Lämpötilan seurantajärjestelmä

Tuukka Lehto

Opinnäytetyö
Maaliskuu 2013
Tietotekniikka
Sulautetut järjestelmät

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Sulautetut järjestelmät ja elektroniikka

TUUKKA LEHTO:
Lämpötilan seurantajärjestelmä

Opinnäytetyö 50 sivua, josta liitteitä 10 sivua
Maaliskuu 2013

Tämä työ tehtiin Tampereen ammattikorkeakoulussa tietotekniikan koulutusohjelmassa. Opinnäytetyön tavoitteena oli puulämmitteisen talon lämmityskeskuksen seurannan helpottaminen ja parantaminen.

Työn suunnittelussa ja toteutuksessa tutustuttiin puulämmitteisen talon tekniseen tilaan, lämmitysjärjestelmän toimintaan, Arduino-kehitysympäristöön, tiedonsiirtotekniikoihin ja työssä vaadittaviin antureihin sekä niiden ohjauspiireihin. Lisäksi huomioitavia asioita olivat laitteiston laajennettavuus, helppokäyttöisyys ja ohjelmakoodin ymmärrettävyys.

Työssä suunniteltu ja toteutettu järjestelmä onnistui tehtävässään hyvin ja sen toiminta on riittävän tarkka käyttökohteeseen. Korjauskertoimien ja veden virtausnopeuksien hienosäädöllä järjestelmästä saadaan hyvinkin tarkka. Järjestelmän luotettavuutta ja kestävyyttä voidaan lisätä piirilevyjen suunnittelulla ja teettämisellä.

Jatkokehittämistä varten laitteistoon on helppo lisätä uusia toiminnallisuuksia muun muassa aurinkokeräimet, sähkönkulutus, WEB-käyttö, kodin seuranta ja monia muita tarpeellisia toiminnallisuuksia.

Asiasanat: arduino, tietoliikenne, anturi, sulautetut järjestelmät

ABSTRACT

Tampere University of Applied Sciences
Information Technology
Embedded Systems

Tuukka Lehto:
Temperature monitoring system

Bachelor's thesis 50 pages, appendices 10 pages
March 2013

The purpose of this thesis was to design a system to improve and ease the temperature monitoring of heating system that uses wood as fuel.

Designing and implementing monitoring system required knowledge of a wood-fueled heating system, the actual heating operation, the Arduino development environment, communication technologies, as well as the sensors and their controllers. In addition, points to consider were the hardware scalability, ease of use and understandability of program code.

The designed and implemented monitoring system succeeded in its task well, and its performance was sufficiently accurate for its purpose. Fine tuning of correction factors and water flow rates will provide very accurate readings from the system. Reliability and durability of the system can be enhanced by designing unique printed circuit board layout which combines all the needed functions of existing PCB's and thus eliminate excessive jump wirings.

In further development it is easy to add new functionalities in the system such as solar power consumption, WEB-use, home security, and many other functionalities.

Key words: arduino, telecommunications, sensor, embedded system

SISÄLLYS

1 JOHDANTO

Tämä työ on tehty Tampereen ammattikorkeakoulun sulautettujen järjestelmien suuntautumisen opinnäytetyönä.

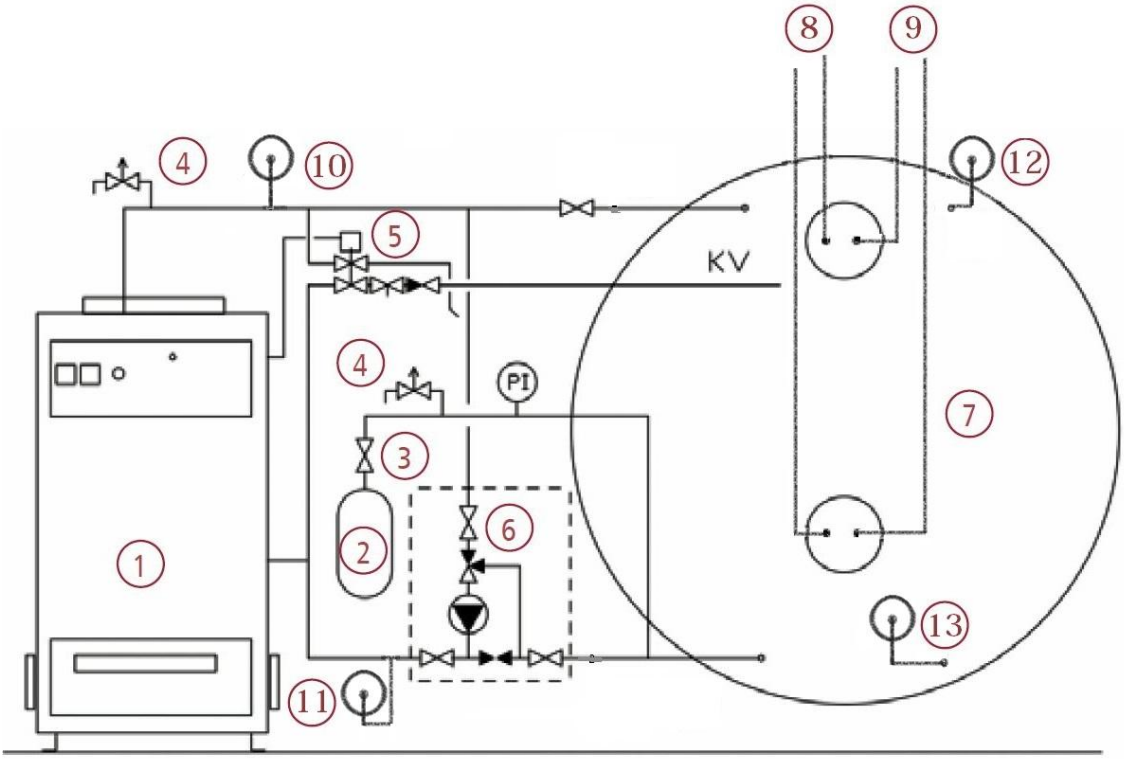
Idea työhön tuli puulämmityksen hankalasta seurannasta. Käytettäessä puuta talon lämmitysmuotona, säädökset estävät lämmityskattilan helpon sijoituksen talossa. Lämmityskattila sijaitsee kohdetalon toisessa päädyssä olevassa teknisessä tilassa, joka yhdistyy kylmän tilan kautta varsinaiseen asuintaloon. Teknisessä tilassa on 20 kW lämmityskattila, 3000 litran vesivaraaja, paisuntasäiliö, putkistot ja pumput. Lämmitettäessä täytyy usein käydä katsomassa savukaasujen, varaajan sekä latauksen lämpötilat. Näiden seurantaa varten täytyi aina kulkea n. 60 metriä kylmän ja likaisen tilan lävitse päästäkseen katsomaan, että kaikki oli kunnossa. Suunnitellulla järjestelmällä voidaan suoraan halutussa paikassa olevasta näytöstä seurata kaikkein tärkeimpiä arvoja, joita ovat savukaasut, säiliön ala- ja ylälämpötilat sekä latausveden pannuun menevän ja sieltä tulevan veden lämpötilat.

Tulevaisuudessa järjestelmää voidaan myös laajentaa uusiin käyttötarkoituksiin mm. aurinkolämmitystä, maalämpöä ja tuulienergiaa varten. Se on suunniteltu siten, että laajentaminen on mahdollista ilman kokonaan uudelleen rakentamista.

2 JÄRJESTELMÄN SUUNNITTELU

2.1 Puulämmitysjärjestelmä

Puulämmitysjärjestelmän pääkomponentteihin kuuluvat lämmityskattila, paisunta-astia, huoltosulkuventtiili, varoventtiili, terminen varolaite, latausyksikkö sekä lämminvesivaraaja. Kytkentä suoritettiin kuvan 1 mukaan.



1. Puulämmityskattila
2. Paisunta-astia
3. Huoltosulkuventtiili
4. Varoventtiili
5. Terminen varolaite
6. Latausyksikkö
7. Lämminvesivaraaja
8. Patterivesi
9. Käyttövesi
10. Kuuman latausveden lämpötila-anturi
11. Kylmän latausveden lämpötila-anturi
12. Varaajan ylälämpötila anturi
13. Varaajan alalämpötila anturi

KUVA 1. Lämmitysjärjestelmän kytkentäkaavio (Arimax puulämmitys, muokattu)

Lämmityskattila on koko järjestelmän sydän. Puun polttamisessa tuotettu lämpöenergia siirretään veteen, jota kierrätetään järjestelmässä niin, että kylmävesi syötetään alhaalta ja kuumavesi lähtee ylhäältä. Tarkoitus on pitää lähtevä kuumavesi noin 80˚C, jotta kattilan sisäosat eivät karstottuisi ja näin ollen heikentäisi kattilan hyötysuhdetta.

Kun vettä kuumennetaan, se laajenee. Koska vesi ei puristu kasaan, rikkoisi se kaikki komponentit lämmitysjärjestelmässä laajentuessaan. Järjestelmää ei myöskään saada täysin ilmattomaksi, joten kuuma ilma laajenee ja näin nostaa järjestelmässä vallitsevaa painetta. Paineen tasaamiseksi käytetään paisunta-astiaa. Tämän astian sisällä on puolivälissä kalvo tai pallo, joka venyy ja antaa myöden paineen kasvaessa järjestelmässä. Päinvastainen tilanne syntyy, kun kalvo tai pallo työntää vettä paineen pitämiseksi tasaisena, jos paine järjestelmässä laskee alhaisemmaksi kuin on tarkoitus. Paisunta-astian liikkuvat osat kuluvat ja sen takia se on mahdollista irroittaa järjestelmästä huoltosulkuventtiilin avulla vaihtoa tai huoltoa varten. Jos taas jostain syystä paine järjestelmässä kasvaa liian korkeaksi, päästää varoventtiili ylimääräisen veden teknisen tilan viemäriin.

Terminen varolaite on pakollinen puukattiloilla. Jos kattilan veden lämpötila nousee 97 ˚C:een, varolaite jäähdyttää kattilan sisällön kylmällä vedellä. Näin ollen ei kiehumisvaaraa ole, koska lämpötila on suurimmillaan 97 ˚C. Tällöin ei myöskään ole höyryn purkaantumisvaaraa varoventtiilistä.

Latausyksikön tarkoitus on pitää kattilasta tulevan latausveden lämpötila mahdollisimman lähellä 80 ˚C:tta. Kun kattila sytytetään, latausyksikkö kierrättää vettä vain kattilassa, jolloin veden lämpötila nousisi mahdollisimman nopeasti 80 ˚C:een. Kun veden lämpötila nousee noin 80 ˚C:een, alkaa latausyksikkö ottamaan vettä myös varaajasta. Näin kattilaan menevän latausveden lämpötila laskee ja yhtä suuri osa kuumasta kattilasta tulevasta vedestä menee varaajaan. Latausvesien lämpötilaa mitataan kuvan 1 antureilla 10 ja 11. Koska vesi kiertää näissä mittauskohdissa vakio virtausnopeudella, voidaan lämpötilaerosta laskea latausteho.

Varaajan tehtävänä on varastoida kattilan tuottama lämpöenergia 3000 litraan vettä. Kun vettä on paljon, syntyvät varaajan sisään kerrokset kylmälle ja lämpimälle vedelle. Kattilasta tuleva lämminvesi saapuu varaajan yläosaan ja vastaavasti kylmää vettä otetaan varaajan alaosasta. Tällöin varaajan vesi ei sekoitu ja vaikka varaajan ylälämpötila näyttää 80 ˚C:tta, niin alapäässä voi olla vasta 30˚C:sta vettä. Tämän takia pitää seurata molempia lämpötiloja, koska vasta alhaalla olevankin veden lämpötilan saavuttaessa 80 ˚C:tta, on varaajan lämmittäminen lopetettava. Varaajassa itsessään on paikat kahdelle lämpötila-anturille. Anturit ovat mekaanisia viisarinäyttöjä, joissa mittapää on upotettu syvälle noin 30 cm:n päähän varaajan ulkokuoresta. Koska

mittapaikkoja ei ole enempää eikä mekaanisia mittarerita haluta ottaa pois käytöstä, on vaihtoehtona asettaa digitaaliset lämpötila-anturit kyseisten mekaanisten vierelle ja eristää tila mahdollisimman hyvin, ettei lämpö karkaa anturipaikasta.

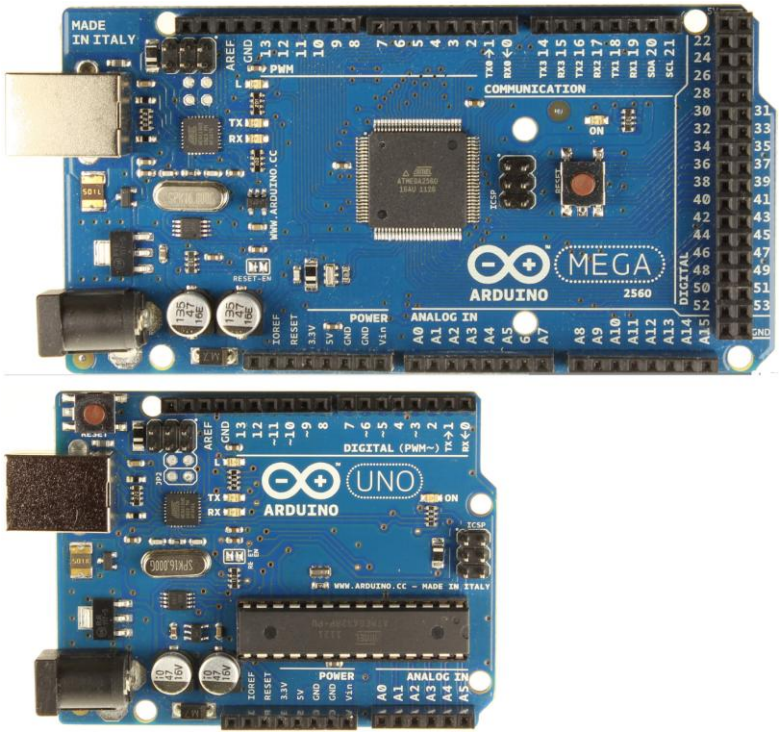
Lämpöpattereiden ja kuuman käyttöveden vesi kuuluvat omiin järjestelmiinsä. Varaajassa on näille omat kuparikierukat, joissa pattereiden ja kuuman käyttöveden vedet kiertävät. Tämä on turvallisuuden ja käyttötarkoituksen kannalta järkevin tapa toteutukselle.

2.2 Arduino-kehitysalusta

Arduino-kehitysalustat kuuluvat avoimen lähdekoodin piiriin. Tämä tarkoittaa sitä, että kuka tahansa voi käyttää, muokata ja kehittää kyseisiä kehitystyökaluja haluamikseen. Koska Arduino kuuluu Creative Commons Attribution-, GPL- ja LGPL-lisenssien alaisuuteen, on kirjastoihin tehdyt muutokset julkaistava kaikkien nähtäville. Muuten omaa ohjelmakoodia tai laitteen sisällä käytettyä laitekorttia ei tarvitse julkaista. Tässä työssä tarkoituksena ei ole kaupallinen hyötyminen, joten kaikki tekemäni työ on muille nähtävissä.

Arduino valmistaa tällä hetkellä yhtätoista Arduino-kehitysalustaa ja viittä shieldiä eli lisälevyä, joita voidaan kytkeä suoraan Arduinon kehitysalustaan ilman juottamista.

Työssä käytettiin kahta Arduino-kehitysalustaa: Unoa ja Megaa. Mega valittiin järjestelmän kordinaattoriksi hyvän laajennettavuuden takia. Koska Megalla ohjattiin myös näyttöä, joka vaati yhteensä kaksikymmentä data- ja ohjauslinjaa, oli käytännössä kaikki pienemmät kortit poissuljettuja. Uno lähetti dataa Mega-kortille käyttäen sarjaliikenneväylää. Alunperin suunnitelmana oli käyttää pelkästään Megaa, josta sitten vedettäisiin johdot antureille pannuhuoneeseen, mutta suunnitelma kariutui muiden tiedonsiirtotekniikoiden epävarmaan toimintaan pitkillä etäisyyksillä. Tämän takia hankittiin Uno-kortti pannuhuoneeseen keräämään dataa sensoreilta ja lähettämään tiedot eteenpäin Mega-kortille niitä pyydettäessä.



KUVA 2. Arduino Mega (ylempänä) ja Uno (Arduino.cc, muokattu)

Arduinon kehitysalustoille on suunniteltu oma ohjelmointityökalu. Arduino-ohjelmointikieli perustuu Wiring-kieleen, joka kuuluu myös avoimen lähdekoodin piiriin ja jolla voidaan ohjelmoida eri laitteistoalustoja mm. AVR atmega, AR Xmega, AVR Tiny, TI MSP430, Microchip PIC24/32-sarja sekä STM M3 ARM -kontrollereita. Ohjelmointikielenä toimii C/C++, jossa Arduino IDE:n avulla on mahdollista käyttää useita valmiita funktioita, jotka helpottavat varsinkin aloittelijan ohjelmointityötä. Ohjelmoinnissa on myös mahdollista käyttää konekielisiä käskyjä piilottamalla ne C-koodin sekaan. (Jämsä Lauri, Arduino-alustat esittelyssä)

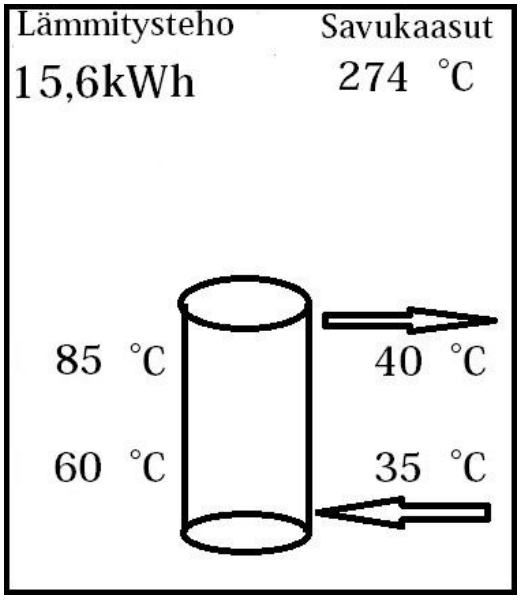
Arduinon ohjelmointityökalu on hyvin pelkistetty, mutta toimiva. Siitä löytyvät kaikki tärkeimmät ominaisuudet, joita ohjelmoinnissa tarvitaan: koodin tarkistus, ohjelmointi mikrokontrollerille, uusi työ, tallennus, lataus ja sarjaliikennemonitori. Asetuksista saa valita oikean COM -portin ja Arduino-kehitysalustan. Kaiken muun ohjelma hoitaa itse. Ohjelman suurimpia kompastuskiviä ovat virheistä tai huomautuksista ilmoittamatta jättäminen sekä simulointi. Niinkuin kaikissa korkean tason kielissä, myös C++:ssa on ongelmana käännöksen tehokkuus ja halutun toiminnon puuttuminen. Vaikka kääntäjä ei virheitä aina annakkaan, ei se tarkoita etteikö siinä virheitä ole. Monet asiat pystytään hoitamaan C++:lla, mutta kaikkea sekään ei osaa halutulla tavalla tehdä. Simulointi Arduino-kehitysalustoilla tehdään ajamalla koodi kehitysalustaan, joka sisältää

sarjaliikenteeseen kirjoitettavia tietoja tutkittavista muuttujista ja tietoa missä kohdassa koodia mennään. Tapa, jolla Arduinolla etsitään koodista vikoja, ei ole vastaavien kilpailijoiden tasolla ja voi aiheuttaa suuriakin ongelmia virheenkorjauksessa, jotka vaatisivat koodin läpikäymistä simuloimalla rivi riviltä.

2.2.1 Käyttöliittymän suunnittelu

Näyttö on olennaisimpia osia tästä työstä. Sen tarkoitus on olla mahdollisimman helppolukuinen ja siitä pitää nopealla vilkaisulla saada tärkeimmät tiedot selville. Tärkeää on myös näytön sijoittaminen paikkaan, jossa se on hyvin saatavilla. Näytön ei tarvitse olla interaktiivinen ainakaan tässä vaiheessa, joten kosketusnäyttöominaisuudet eivät ole pakollisia, mutta laajennettavuuden takia niistä ei varmasti haittaakaan ole. Valtaosa nykyajan LCD-näytöistä ovat vähintään 256 väriä toistavia. Tässä työssä esimerkiksi punainen voisi ilmoittaa hälyyttävistä arvoista, mutta kokemukset halpojen LCD-näyttöjen värientoistokyvystä ovat hyvin epämääräisiä ja värien sävyt tummia. Tällöin ne aiheuttaisivat enemmänkin silmien siristelyä kuin varsinaisen tekstin huomaamisen. Tämän takia työssä käytetään vain valkoista tekstiä ja kuvia mustalla pohjalla, jolloin saadaan mahdollisimman suuri kontrasti aikaiseksi. Näytön koko täytyisi olla tarpeeksi iso, jotta siitä näkisi tarvittavat tiedot pelkällä vilkaisulla menemättä lähelle tai silmiä siristämättä. Näyttö ei saisi myöskään olla liian suuri, ettei se veisi huomiota kaikelta muulta, vaan sopisi sisustukseen mahdollisimman hyvin ja huomaamattomasti. Hyvä näyttökoko löytyykin jokaisen taskusta löytyvästä puhelimesta. Se on tarpeeksi iso tekstille, mutta ei ole liian isokokoinen kooltaan, joten työhön valittiin yleisin puhelimen näyttökoko 3,2".

Näytön perusnäkymän suunnittelussa tärkeintä oli yksinkertaisuus ja selvyys. Näytöstä pitäisi nähdä latausvesien, säiliön ja savukaasujen lämpötilat sekä lämmitysteho. Ensimmäinen vedos sisälsi pelkkää tekstiä, joka kumminkin osoittautui hankalalukuiseksi. Kaikkien tietojen listaaminen tekstinä alekkain hankaloitti näkymän selkeyttä. Näytöltä ei ollut helppoa katsoa yhtä arvoa, vaan täytyi "lukea" kaikki alusta loppuun, koska oikean kohdan löytäminen oli vilkaisulla vaikeata. Yksinkertaisen grafiikan lisääminen, esimerkiksi nuolet, helpottaa oikean tiedon löytämistä, kunhan sijoittelu on järkevä. Suunnitelu tehtiin MS Paint -ohjelmalla lopullisen mallin näyttäessä kuvan 3 mukaiselta, jonka kanssa siirryttiin toteutusvaiheeseen.



KUVA 3. Suunnitteluvaiheen näytön perusnäkymä.

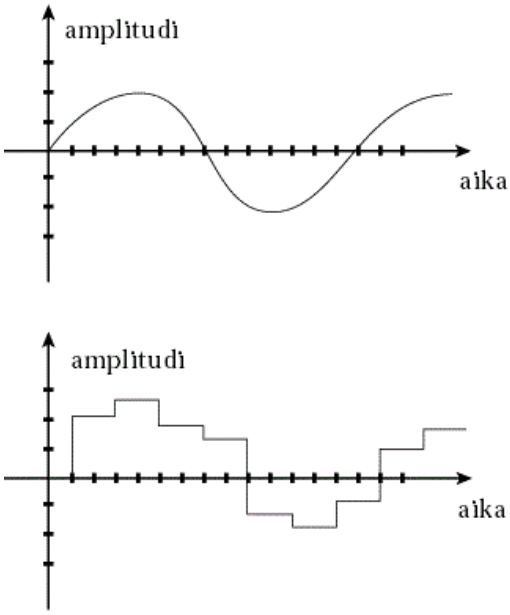
2.3 Tietoliikenne

Tietoliikenteellä tarkoitetaan tiedon siirtoa lähettäjältä vaastaanottajalle siirtotietä pitkin. Siirtotie voi olla langallinen tai langaton. Nykyaikana suositaan kaikissa laitteissa langattomuutta, mutta tässä järjestelmässä sillä ei saavutettaisi mitään hyötyöä langalliseen verrattuna, joten työ tehtiin langallisesti. Tieto, joka siirtotiessä liikkuu, voi olla sekä analogista että digitaalista. (Signaalinkäsittelytekniikan laboratiorio)

Analogisessa tiedonsiirtotekniikassa signaalilla on määrittämätön määrä eri arvoja, jolloin analogisen signaalin datamäärä voi olla paljon suurempi kuin digitaalisen. Ongelmana on analogisen tiedon vaimeneminen ja vääristyminen sekä häiriön vahvistaminen signaalia vahvistettaessa. Tämän takia analoginen signaali ei ole kovin luotettava suuria datamääriä sisältävän datan lähettämiseen. Yksinkertaisuutensa vuoksi sitä kumminkin käytetään monissa eri sovelluksissa. (Signaalinkäsittelytekniikan laboratorio)

Digitaalinen signaali perustuu täsmällisiin arvoihin. Yleensä ja yksinkertaisimmassa sovellusmallissa se voi saada aina vain arvot yksi ja nolla. Tätä kutsutaan binäärijärjestelmäksi. Tällöin esimerkiksi viiden voltin jännite voidaan jakaa kahteen eri

loogiseen tilaan. Nollasta kahteen ja puoleen volttiin tarkoittaa loogista nollaa ja taas kahdesta ja puolesta voltista viiteen volttiin tarkoittaa loogista ykköstä. Digitaalisella signaalilla voi olla myös useampia arvoja, mutta ei koskaan niin useita kuin analogisessa. Aina kun analogista signaalia muutetaan digitaaliseksi, katoaa muutoksessa dataa. A/D-muunnoksessa pahimmassa tapauksessa signaali laskostuu, jos näytteenottotaajuus on liian alhainen ja näin saatu signaali on käyttökelvoton. Digitaalisen signaalin korjaus on helpompaa kuin analogisen, koska digitaalisen signaalin ennustettavuus on parempaa. Digitaalisessa signaalinkäsittelyssä käytetäänkin paljon virheenkorjausta, jotta data olisi mahdollisimman luotettavaa. Siirtojohtimen häiriöt eivät haittaa digitaalista signaalia niin paljon kuin analogista. Digitaalisella signaalilla on vain kaksi arvoa. Tämän takia ei pienillä jännitteen heittelyillä ole merkitystä loogisen tilan lukemiseen, ellei jännitteen heittely tapahdu juuri loogisten tila-arvojen välillä, eli esimerkkitapauksessa noin kahden ja puolenvoltin kohdalla. (Signaalinkäsittelytekniikan laboratiorio)

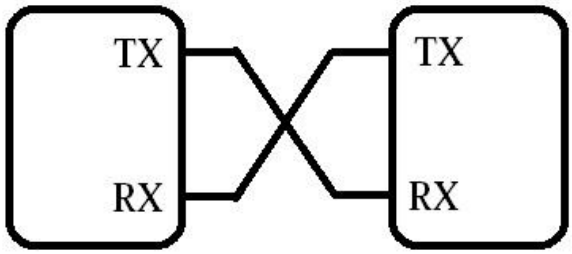


KUVA 4. Analoginen signaali ylhäällä ja sitä vastaava digitaalinen signaali alhaalla (Considerable Sounds, muokattu)

Hyvänä esimerkkinä analogisen ja digitaalisen signaalin eroista nähdään nykyajan televisiovastaanottimissa. Analogisten lähetysten aikana heikolla signaalin voimakkuudella kuvaan tuli ns. lumisadetta. Digitaalisessa lähetyksessä kuva näkyy tiettyyn pisteeseen asti täydellisenä, mutta kuva katoaa televisiosta täysin signaalin alittaessa virheenkorjauksen rajakynnyksen.

2.3.1 Sarjaliikennetekniikka ja UART-piiri

Sarjaliikennetekniikan standardi RS232 on synkronoimaton tai synkronoitu sarjaliikennetekniikka. Sana sarjaliikenne tarkoittaa, että jokainen bitti lähetetään yksitellen ja peräkkäin toisin kuin rinnakkaisliikenteessä, jossa kaikki bitit lähetetään samaan aikaan rinnakkaisilla johtimilla. Sarjaliikennetekniika mahdollistaa saman datamäärän siirtämisen huomattavasti vähemmällä johtomäärällä kuin rinnakkaisliikennetekniikka. Näin järjestelmän hinta laskee ja luotettavuus paranee. Synkronoimattomalla tarkoitetaan bittien lähettämistä peräkkäin, ilman että niille on annettu tietyt aikakanavat. Tämän synkronoimattoman liikenteen etuina ovat johtimien vähäisempi määrä, synkronointibittien puuttuminen ja tästä johtuva suurempi kaistanleveys sekä signaalin parempi kantomatka. Synkronoitua liikennettä ei useimmiten käytetä mikrokontrollereissa, mutta sen erityinen etu synkronoimattomaan nähden on aloitusbitin tunnistaminen oikeassa kohdassa lähetettä ja viestiä ei lueta virheellisesti tämän takia. Johtimia synkronoimattomassa tekniikassa on kaksi sekä yhteinen maajohdin. Sarjaliikennedata kulkee johtimessa vain yhteen suuntaan. Jotta viestit kulkisivat molempiin suuntiin, vaatii se kaksi johdinta. Kuvasta 5 nähdään synkronoimattoman kytkentä. (Lammert Bies, RS232 Specifications and standard)



KUVA 5. Sarjaliikennekytkentä kahden laitteen välillä.

Sarjaliikenteessä data lähetetään yleisesti kahdeksan bitin jaksoissa, eli tavuissa. On myös mahdollista valita viiden ja kahdeksan bitin väliltä, mutta useimmiten käytössä on kahdeksan bittiä. Käytettäessä synkronointia tai virheenkorjausta, vievät nämä bitit tilaa varsinaiselta datalta lähetteestä. Tällöin ei voida käyttää kahdeksaa databittiä, mutta näitäkään ei useimmiten tarvita tai käytetä mikrokontrollerisovelluksissa. Viestin lähettäjällä ja vastaanottajalla pitää olla asetettuna sama arvo lähetteen pituudelle, jotta

viesti ymmärretään oikein vastaanottopäässä. (Lammert Bies, RS232 Specifications and standard)

Tiedonsiirtonopeus (baud rate) on valittavissa useammista vaihtoehdoista. Arduino tukee nopeuksia: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 ja 115200 bittiä sekunnissa. RS232-standardi tukee nopeuksia aina 20 kbps asti, vaikka nopeimpia siirtonopeuksia ei kaikissa laitteissa tuetakkaan. Nopeus kannattaakin valita mahdollisimman lähelle vaadittua miniminopeutta, koska sillä on suora vaikutus maksimaaliseen johtimen pituuteen. Texas Instrumentin tutkimuksen mukaan UTP CAT-5 kaapelilla tyypillinen maksimi kaapelipituus on yleisimmillä 2400-19200 bps nopeuksilla taulukon 1 mukaiset. (Lammert Bies, RS232 Specifications and standard)

TAULUKKO 1. Nopeuden vaikutus kaapelin maksimi pituuteen. (Lammert Bies, RS232 Specifications and standard, muokattu)

Bittiä sekunnissa	**Maksimi kaapelin pituus (m)**
19200	15
9600	150
4800	305
2400	915

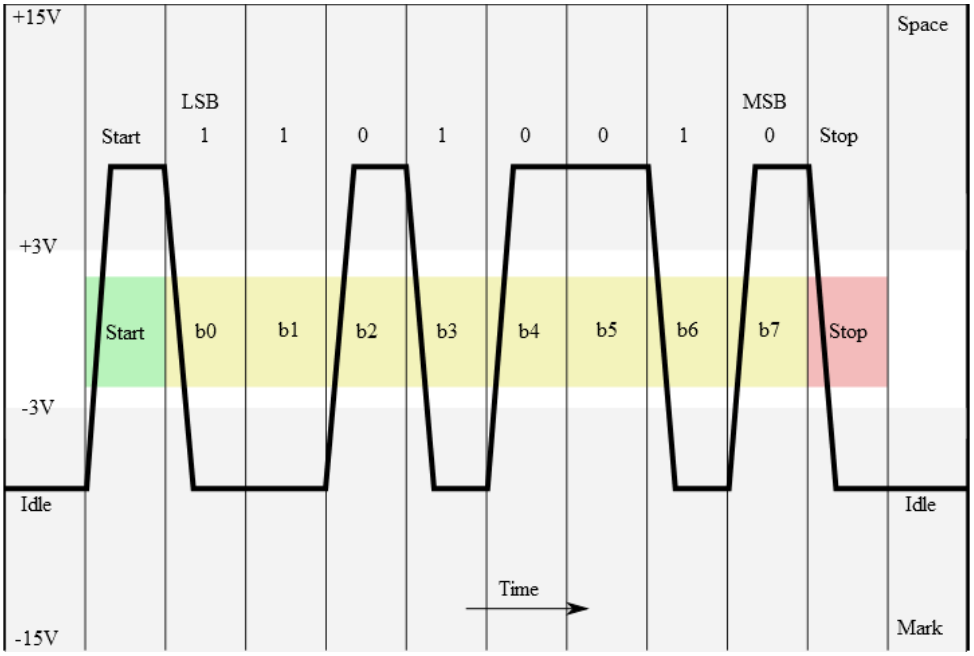
Sanoma aloitetaan aloitusbitillä. Synkronoimattomassa sarjaliikenteessä lähete voi tulla vastaanottopäähän milloin tahansa. Se voi aiheuttaa ongelmia vastaanotossa ensimmäisen bitin tunnistamisessa ja sitä varten on lisätty jokaisen sanoman alkuun aloitusbitti eli huomiobitti. Tällöin linja menee ylös, jotta vastaanottopää selvästi erottaa lähetteen alkavan. (Lammert Bies, RS232 Specifications and standard)

Databitit tulevat heti aloitusbitin jälkeen. Looginen ykkönen vastaa linjan alatilaa ja looginen nolla taas linjan ylätilaa. Databiteistä vähiten merkitsevä lähetetään ensimmäisenä. (Lammert Bies, RS232 Specifications and standard)

Databittejä seuraa pariteettibitti, jota käytetään virheenkorjauksessa. Lähetin laskee pariteettibitille arvon lähetetyistä databiteistä standardin mukaisen kaavan avulla. Kun sanoma saapuu vastaanottajalle, laskee se databiteistä samalla kaavalla pariteettibitin ja vertaa sitä lähettimen antamaan bittiin ja joko hyväksyy sanoman tai hylkää sen virheellisenä. (Lammert Bies, RS232 Specifications and standard)

Viimeisenä sanomassa lähetetään lopetusbitit. Näitä voi olla useampia, mutta yleisimmin käytetään vain yhtä. Sanoman lopussa lähetetään lopetusbitti, joka on aina alatilassa. Jos vastaanottopää odottaa lopetusbittiä ja jos se saa sen kohdalla jonkun muun kuin alatilan, hylätään sanoma ja synkronoidaan uudestaan seuraavalle sanomalle. (Lammert Bies, RS232 Specifications and standard)

Sarjaliikenteen kehys koostuu aloitusbitistä, databiteistä, pariteettibitistä ja lopetusbiteistä. Alla olevasta kuvasta nähdään sarjaliikenteen kehysrakenne. (Lammert Bies, RS232 Specifications and standard)

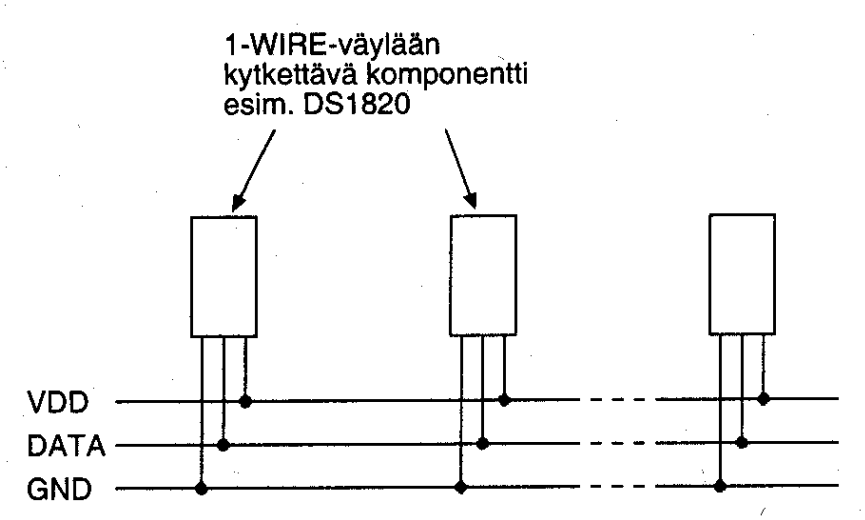


KUVA 6. Sarjaliikenteen kehys (Bluetooth Controlled Led Driver)

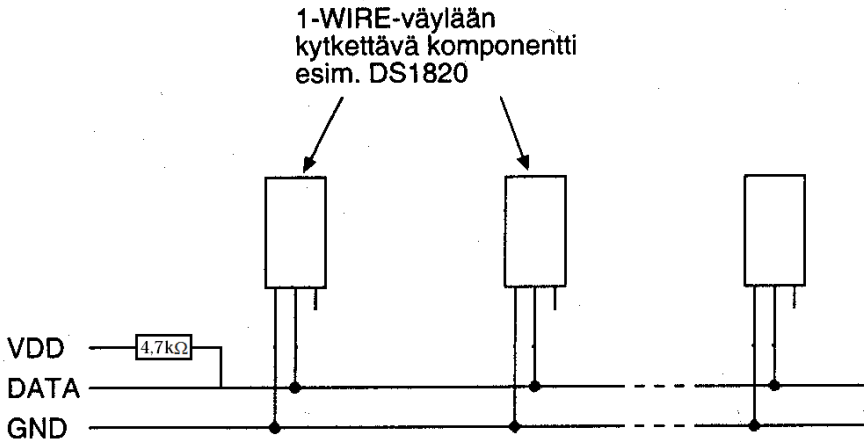
2.3.2 1-Wire -tekniikka

1-Wire vaatii toimiakseen kaksi johdinta, vaikka nimi antaisikin ymmärtää, että yksi johdin riittäisi. Niinkuin kaikissa siirtotekniikoissa, tarvitaan myös tässä datajohtimen lisäksi maajohdin. Kuvassa 7 nähdään kuinka komponentit kytketään linjaan, jos myös käyttöjännitteet kytketään. Väylään liitettäviä komponentteja kutsutaan orjiksi ja isäntänä toimii mikrokontrolleri. 1-Wire -tekniikassa liikennöinti on kaksisuuntainen, jolloin isäntä lähettää ja vastaanottaa orjan datan samaa johdinta pitkin tiedonsiirtonopeudella 16,3 kbps ja samassa johtimessa voi olla jopa sata eri laitetta. 1-

Wire -tekniikan tärkeimpiin ominaisuuksiin kuuluu käyttöjännitteen syöttö orjakomponenteille datalinjaa pitkin. Lepotilassa väylä on aina ylätilassa ja vain datan lähettämisen aikana hyvin lyhyitä aikoja alatilassa. Useampia komponentteja kytketäessä linjaan, vaaditaan datalinjalta suuria virtamääriä. Virran riittävyys datalinjassa toteutetaan käyttämällä ylösvetovastusta viiden voltin käyttöjännitteestä datalinjaan. Orjakomponentteihin on rakennettu sisäiset kondensaattorit, jotka varastoivat energiaa datan lähettämisen ajaksi etteivät orjat sammuisi kesken lähetyksen. (1-wire -tekniikka)

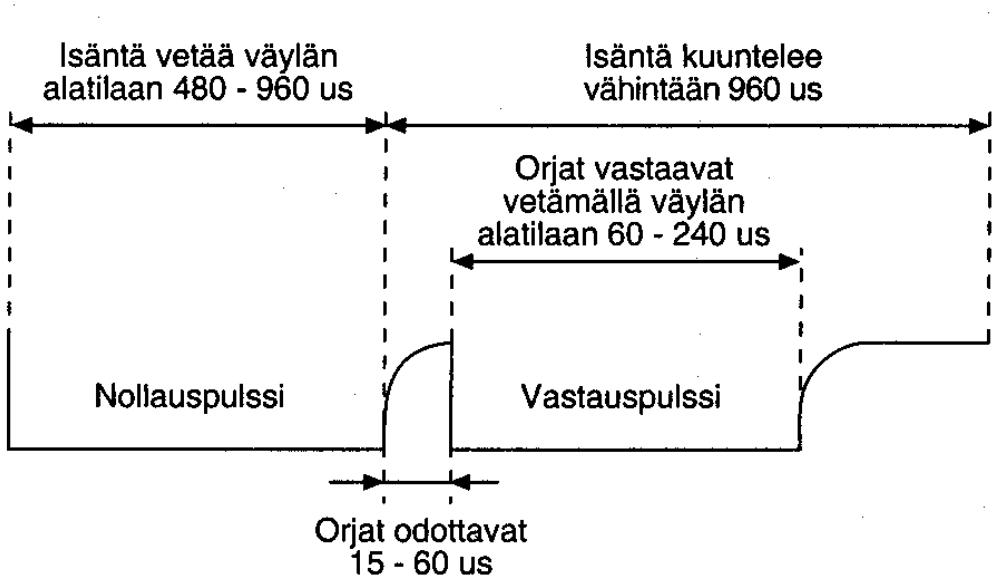


KUVA 7. Komponenttien kytkentä linjaan käyttöjännitelinjan kanssa. (1-wire -tekniikka, muokattu)



KUVA 8. Komponenttien kytkentä linjaan ilman käyttöjännitelinjaa (1-wire -tekniikka, muokattu).

Väylällä vain isäntä voi aloittaa tiedonsiirron. Tiedonsiirto alkaa, kun isäntä lähettää linjaan nollauspulssin, jolloin isäntä vetää linjan alatilaan 480 - 960 µs ajaksi ja tämän jälkeen isäntä alkaa kuunnella väylää. Väylällä olevat orjat vastaavat nollapulssiin vetämällä väylän alatilaan 15 - 60 µs kuluttua isännän nollauspulssista 60 - 240 µs ajaksi kuten kuvasta 9 nähdään. (1-wire tekniikka)



KUVA 9. Nollauspulssin aikakaavio (1-wire -tekniikka).

Jokaisen kirjoituspulssin välissä pitää olla vähintään 1 µs tauko. Isännän vetäessä linja**n** alas pitemmäksi kuin 15 - 60 µs ajaksi, lukee orja kyseisen bitin nollana. Väylän palattaessa takaisin ylätilaan 15 - 60 µs jälkeen, lukee orja bitin ykkösenä. Isännän lukiessa dataa orjasta, lähettää se lukupulssin vetämällä väylän alatilaan vähintään yhdeksi mikrosekunniksi ja antaa tämän jälkeen ylösvetovastuksen nostaa väylä ylätilaan. Tähän orja vastaa heti asettamalla väylän alatilaan seuraaviksi 15 µs, kun kyseessä on nollabitti ja vastaavasti ykkösbitillä antaa väylän olla ylhäällä seuraavat 15 µs. (1-wire -tekniikka)

2.3.3 SPI-tekniikka

Motorolan kehittämä sarjaliikenneväylä (Serial Peripheral Interface). SPI-väylä käyttää neljää signaalijohdinta. Kellosignaali, MOSI (Master Output Slave Input), MISO (Master Input Slave Output) ja SS (Slave Select). MOSI-linjan kautta isäntä lähettää tietoa orjalaitteille ja MISO-linjan kautta tieto kulkee orjilta isännälle. Tämä mahdollistaa full-dublex -tekniikan käytön tiedonsiirrossa. Eli isäntä voi samaan aikaan sekä lähettää että vastaanottaa orjalta dataa. SPI-väylätekniikassa ei ole osoitteita. Haluttu orja valitaan vetämällä orjan SS-linja alatilaan. Valinnan jälkeen aloitetaan tiedonsiirto kellopulssista. Kellopulssilla voidaan valita siirtyykö data nousevalla vai laskevalla reunalla. Jokaisen kellopulssin aikana siirtyy yksi bitti. Kun kahdeksan bittiä on lähetetty, nousee SS-linja ylätilaan. SPI-väylän kehysrakenne on samanlainen kuin Serial-väylän sillä erotuksella, että looginen ykkönen on ylätila ja nolla volttia on lepotila. (Asko Olli, Sulautettujen järjestelmien väylätekniikat)

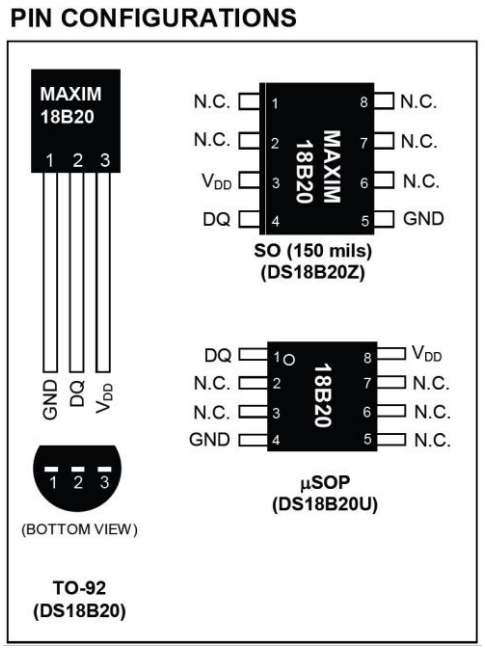
2.4 Anturit

Anturit ovat laitteita, jotka mittaavat ympäristöstään haluttuja suureita ja muuttavat ne mittalaitteille helpommin ymmärrettävään muotoon. Mitattavina suureina voivat olla mm. lämpötila, paine, nopeus, paino jne. Antureilla on paljon ominaisuuksia jotka vaikuttavat mittaustuloksiin joko halutusti tai virheellisesti. Täydellistä anturia ei olekaan ja tämän takia monissa antureissa pitääkin ottaa huomioon mm. epälineaarisuus, hystereesi, resoluutio, erotuskynnys, toistettavuus ja mittausepävarmuus. (H. Honkanen, Anturit)

2.4.1 Dallas DS18B20-lämpötila-anturi

Dallas Semiconductor Corporation perustettiin vuonna 1984 ja saavutti nopeasti johtavan aseman puolijohteiden ja mikrosirujärjestelmien valmistajana. Maxim Integrated Prodictions kuitenkin osti Dallasin vuonna 2001 ja uudemmat tuotteet käyttävätkin Maximin nimeä, mutta vanhemmat mallit kuten tämä lämpötila-anturi käyttävät vielä vanhempaa Dallasin nimeä tunnistettavuutensa takia. (Maxim Integrated Products Inc 2012)

Dallas DS18B20 on digitaalinen 1-wire -tekniikalla toimiva lämpötila-anturi neljällä eri paketointivaihtoehdolla (kuva10) ja säädettävällä 9, 10, 11 tai 12-bitin mittaustarkkuudella, joita vastaavat lämpötilatarkkuudet ovat 0,5 ˚C, 0,25 ˚C, 0,125 ˚C tai 0,0625 ˚C. Sen pääkomponentit ovat 64-bit laaserpoltettu ROM, varsinainen lämpötila-anturi ja haihtumaton muisti hälytyksille (TH ja TL). (Maxim Integrated DS18B20 datasheet)



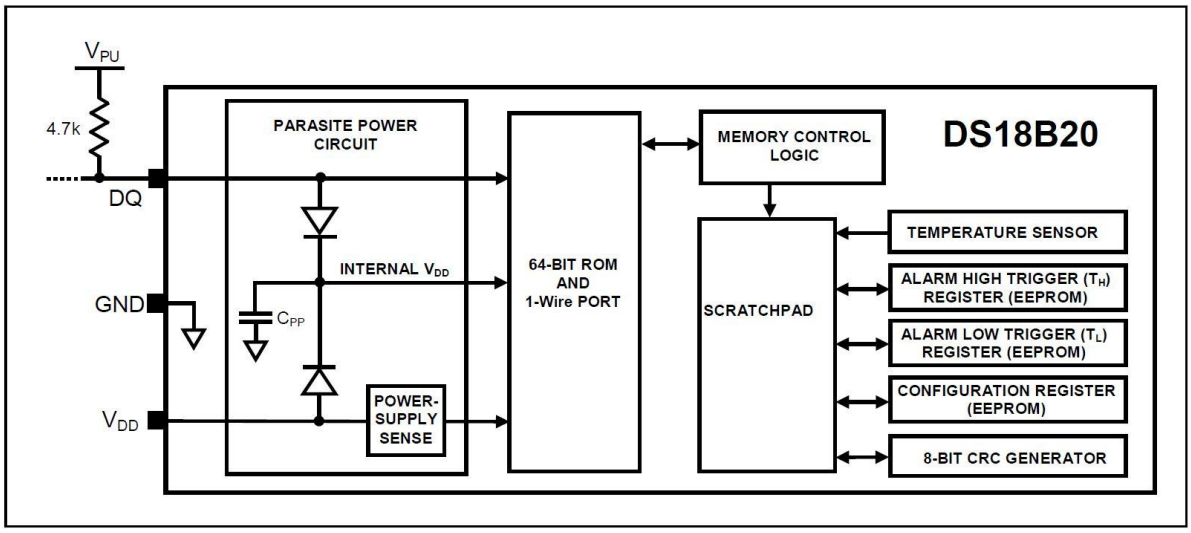
KUVA 10. DS18B20-lämpötila-anturin pinnijärjestys ja kotelointivaihtoehdot (Maxim Integrated DS18B20 datasheet)

Anturin 48 bittinen sarjanumero on yksilöllinen, eikä yhdelläkään toisella ole samanlaista. Tätä sarjanumeroa käytetään osoitteena, josta tiedetään mikä anturi on kyseessä, kun linjasta kysytään lämpötilaa. Sarjanumero mahdollistaa useiden DS18B20 antureiden kytkemisen yhteen linjaan, mutta käytännössä niiden määrää rajoittaa enemmänkin mittauksien hitaus kuin anturien määrä. Muistialueen ensimmäiset kahdeksan bittiä sisältävät CRC eli tarkistusbitit ja viimeiset kahdeksan bittiä laiteperhetunnisteen. ROM-muistin rakenne kuvassa 11. (Maxim Integrated DS18B20 datasheet)

8-BIT CRC		48-BIT SERIAL NUMBER		8-BIT FAMILY CODE (28h)	
MSB	LSB	MSB	LSB	MSB	LSB

KUVA 11. ROM-muistin rakenne (Maxim Integrated DS18B20 datasheet).

Anturin toimintaperjaate perustuu piioskilaattorin värähtelytaajuuteen. Värähtelytaajuus muuttuu lämpötilan mukaan, jolloin laskuri laskee värähtelyjen määrän tietyllä aikajaksolla ja lopuksi tulos korjataan piin epälineaarisen käyttäytymisen takia. (Dallas Semiconductor DS1820)



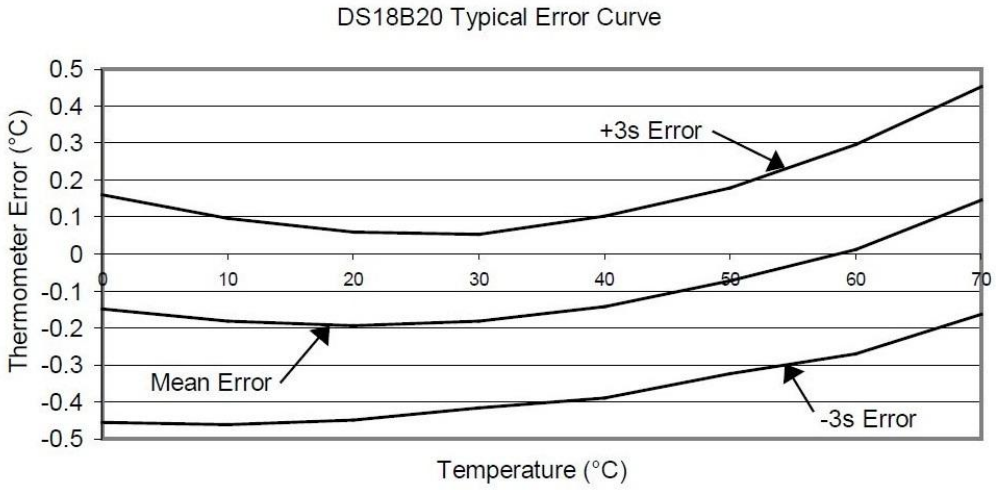
KUVA 12. DS18B20 lämpötila-anturin lohkokaavio. (Maxim Integrated DS18B20 datasheet)

Hälyytykset saadaan asetettua anturiin kirjoittamalla työmuistin (scratchpad) bittien kaksi (TH) ja kolme (TL) rekistereihin kahdeksan bittiset lämpötila-arvot. Nämä arvot pysyvät muistissa, kunnes toisin kirjoitetaan. Anturin mitattua lämpötilan, se vertailee sitä hälytysrekistereiden arvoihin. Lämpötilan ollessa suurempi kuin TH:n arvo tai pienempi kuin TL:n arvo, hälytyslippu nousee ylös. Hälytyksiä käytettäessä täytyy ottaa huomioon, että jos käytössä on suurempi mittaustarkkuus kuin kahdeksan bittiä, eivät hälytykset ota huomioon mahdollisia neljää lisäbittiä. Tällöin tarkin mahdollinen hälytylämpötila on silti vain ±0,5 ˚C erottelulla. (Maxim Integrated DS18B20 datasheet)

Anturin tärkeimpiin ominaisuuksiin kuuluu parasiittimoodi, jolloin laite ottaa virtansa 1-Wire väylästä. Kun väylän tila on ylhäällä, se lataa sisäiseen kondensaattoriinsa

tarpeeksi energiaa lyhyitä 1-wire väylän lähetystilanteita varten. Parasiittimoodi on todella hyödyllinen sovelluksissa, joissa etäisyydet ovat pitkiä ja tilat ahtaita. Kun käytetään parasiittimoodia, täytyy pinnin VDD olla kytkettynä maihin. Kovassa rasituksessa mm. datan lähetyksessä saattaa anturi viedä jopa 1,5mA virtaa ja parasiittimoodissa täytyykin käyttää vahvaa ylösvetovastusta, ettei jännite tipahda linjassa alle hyväksyttävän rajan. Suurissa lämpötiloissa parasiittimoodia ei suositella, koska anturin toimintavarmuutta ei voida taata 100 ˚C tai sen yli olevissa lämpötiloissa sisäisten ohivirtavuotojen takia, tarvitaan erillistä jännitelähdettä, josta anturille syötetään +5 V käyttöjännite. (Maxim Integrated DS18B20 datasheet)

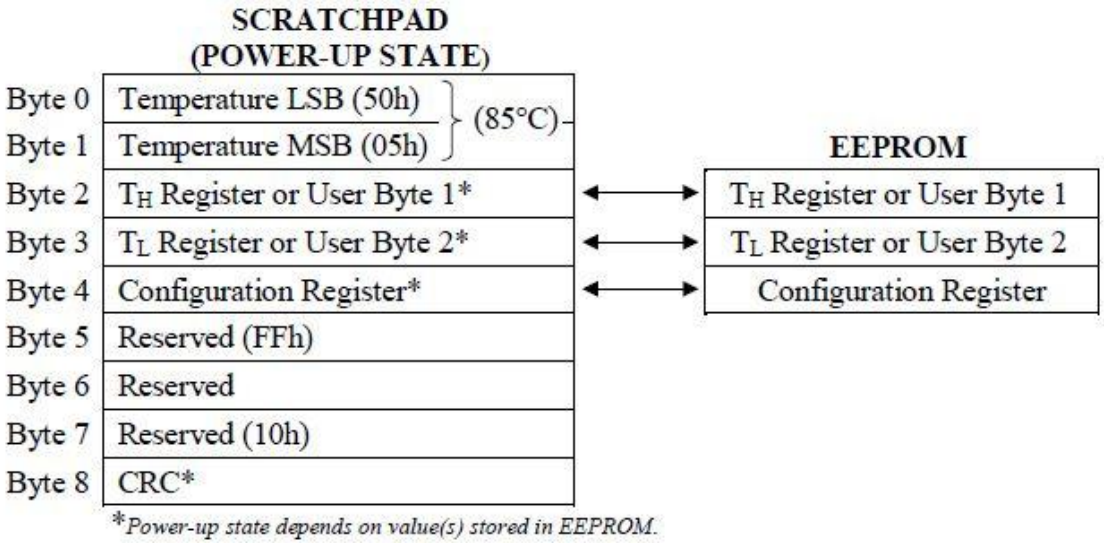
Anturin toimintalämpötila on -55 ˚C aina +125 ˚C asti, mutta tarkempi ±0,5 ˚C virheellä saatava tulos on -10 ˚C ja +85 ˚C välillä. Kuvasta 13 nähdään hieman rajoittunut, mutta ainoa valmistajan tarjoama suorituskykykäyrä (Maxim Integrated DS18B20 datasheet)



KUVA 13. DS18B20 lämpötila-anturin suorituskykykäyrä. (Maxim Integrated DS18B20 datasheet)

Anturin operointi aloitetaan 1-wire -väylän aloituspulssilla (kappaleessa 2.3.2) ja sen jälkeen käyttämällä jotakin seuraavista ROM-kutsuista: ROM:in etsintä(F0h), ROM:in lukeminen(33h), ROM:in täsmäys(55h), ROM:in ohittaminen(CCh) tai hälytyslipun tila(ECh). ROM:in etsintäkutsu lähetetään, jotta selvitetään montako laitetta on kytketty linjaan ja niiden laitetiedot (sarjanumero, CRC ja laiteperhetunniste). Kutsu voidaan joutua lähettämään useita kertoja, jotta varmasti saadaan kaikilta orjalaitteilta vastaus. Jos linjassa on vain yksi laite, voidaan käyttää yksinkertaisempaa ROM:in lukukutsua. Tällä kutsulla luetaan suoraan orjan 64-bittinen ROM:in sisältö, eikä ole

tarvetta käyttää monimutkaisempaa etsintäkutsua. ROM:in täsmäyskutsussa isäntä lähettää täsmäyskutsun ja sen perässä 64-bittisen ROM:in sisällön, johon vain kyseisen ROM:in sisältävä orja vastaa ja näin varmistetaan oikean orjalaitteen löytyminen. Haluttaessa lähettää viesti kaikille orjalaitteille kerralla, voidaan se tehdä ROM:in ohituskutsua käyttäen. Isännän lähettäessä linjaan ohituskutsun ja sen perässä lämpötilapyyntökomennon, alkavat kaikki orjat työstämään lämpötilanmittausta. Ohituskutsua ei kumminkaan saa käyttää työmuistin lukemisfunktion edellä, jos useampi orja on linjassa, koska se aiheuttaa datan törmäysvaaran linjassa kaikkien lähettäessä linjaan dataa samaan aikaan. Jos linjassa kuitenkin on vain yksi laite, voidaan kyseinen kutsu lähettää työmuistin lukemisfunkion edellä. Lähetyslipun tilan kutsu lähetetään linjaan isännän toimesta ja siihen vastaavat vain ne orjat joiden hälytyslippu on ylhäällä viime lämpötilamittauksen toimesta. Lipun tilan rekisteröinnin jälkeen täytyy linjalle tehdä resetointi ja orjat nollattava sekä aloitettava käynnistyssekvenssi alusta. (Maxim Integrated DS18B20 datasheet)

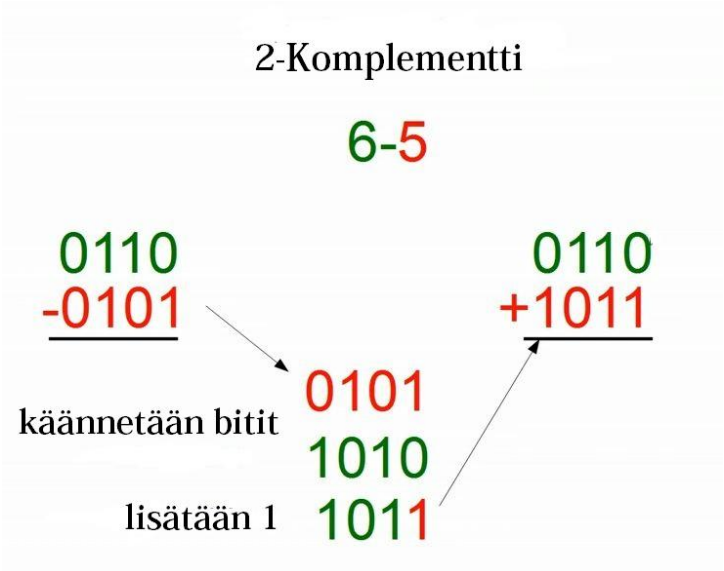


KUVA 14. Työmuistin rakenne. (Maxim Integrated DS18B20 datasheet)

Isännän lähetettyä jonkun ylläolevista ROM-kutsuista, voidaan niiden perään lähettää haluttu funktiokäsky. Näillä funktiokäskyillä isäntä voi kirjoittaa tai lukea anturin työmuistista. Funktiokäskyjä on kuusi kappaletta. Lämpötilamittaus (44h), työmuistiin kirjoitus (4Eh), työmuistin lukeminen (BEh), työmuistin kopiointi (48h), kopioi hälytysarvot ROM:sta (B8h) ja virtalähteen selvitys (B4h). Lämpötilamittausfunktiolla käynnistetään mittauksen aloitus ja muunnos, sekä talletus työmuistin kahteen tavun kokoiseen rekisteriin, jonka jälkeen anturi palautuu lepotilaan. Parasiittimoodin ja

erillisen virtalähteen käyttö eroaa toisistaan komentoa käytettäessä. Erillisvirtalähteen kanssa datalinja menee nollaan, kunnes funktio on suoritettu. Parasiittimoodissa tämä ei ole mahdollista, koska muuten anturilta itseltään ja muilta mahdollisilta antureilta loppuisi virta. Tarkoitus onkin pitää linja ylhäällä koko funktiosuorituksen ajan ja ilmoitusta tuloksen valmistumisesta ei tehdä. Työmuistiin kirjoittamisfunktiolla voidaan kirjoittaa kolme tavua järjestyksessä TH, TL ja asetusrekisteri. Työmuistin lukemisfunktiolla voidaan lukea koko työmuisti läpi alkaen tavun nolla vähiten merkitsevästä bitistä eli lämpötilan vähiten merkitsevimmästä bitistä. Luku voidaan keskeyttää resetillä missä vaiheessa vain toiminnan nopeuttamiseksi. Työmuistin kopiointifunktiolla voidaan kopioida TH-, TL- ja asetusrekisterit ROM-muistille talteen ja toiseen suuntaan taas hälytysarvojen kopioimisfunktiolla ROM:sta. Virtalähteen selvitysfunktion saatuaan, orja vetää datalinjan alas parasiittimoodissa ja jos taas käyttää erillistä virtalähdettä, pitää linjan ylhäällä. (Maxim Integrated DS18B20 datasheet)

Lämpötila saadaan lämpötila-anturilta kahdesta muistirekisteristä LSB ja MSB tässä järjestyksessä. Lämpötila-anturi käsittelee lämpötila-arvon celsiusmuodossa, jolloin myös miinusmerkki täytyy olla käytössä. Tätä varten on kehitetty 2-komplemetti. Negatiiviset luvut voidaan ilmoittaa lisäämällä eteen ns. etumerkkibitti. Jos etumerkki on 1, niin tällöin kyseessä on 2-komplementti ja luvun vastaluku saadaan kääntämällä luvun bitit ja lisäämällä yksi tulokseen, kuten kuvassa 15 nähdään yksinkertaisessa vähennyslaskussa. (Maxim Integrated DS18B20 datasheet)



KUVA 15. 2-komplementti vähennyslasku.

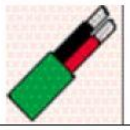
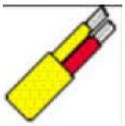
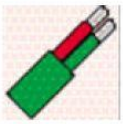
Oletetaan, että lämpötila-anturi lähetti lämpötilan 9-bitin tarkkuudella arvon 1 1011 1001, jonka eniten merkitsevä bitti on 1 ja tarkoittaa miinusmerkkistä tulosta, joten sille tehdään 2-komplementti. Ensimmäisenä käännetään binääriluvut 1011 1001 vastaluvuikseen 0100 0110 johon lisätään yksi, eli 0100 0111 jonka normaali desimaaliesitys on 71. Koska käytämme 9-bittistä tarkkuutta, vähiten merkitsevä bitti merkitsee vain 0,5 celsiusasteen hyppäystä, saa vähiten merkitsevä bitti potenssiksi -1.
$-(0*2^6+1*2^5+0*2^4+0*2^3+0*2^2+1*2^1+1*2^0+1*2^{-1}) = -(32+3+0{,}5) = -35{,}5$ ˚C

2.4.2 Termopari-anturi

Termpopari (Termoelementit, Thermocouples) on lämpötila-anturi, joka on saavuttanut suuren suosion yksinkertaisen rakenteen, halvan valmistushinnan, suuren lämpötilankeston ja tarkkuuden takia. Tarkkuutta onkin pidetty termoparien heikkoutena, mutta nykyään päästään hyvällä suunnittelulla jopa 0,5 ˚C tarkkuuteen. (Termopari lämpötila-anturina)

Termoparien toiminta perustuu Seebeckin ilmiöön, jonka keksi Virolainen fyysikko Thomas Seebeck vuonna 1822. Seebeckin teorian mukaan kahden eri metallin liitokseen syntyy lämpötilasta riippuvainen jännite. Tämä pätee lähes kaikkiin metalleihin, mutta toisilla metalleilla saadaan suurempia jännitteitä lämpötilan funktiona kuin toisilla. Termoparin jännite kuvaa aina kahden pisteen lämpötilaeroa. Kohteessa olevan liitoksen, jossa kaksi eri metallia kohtaavat sanotaan mittapääksi ja kohdassa, jossa termoparin metallit vaihtuvat normaaleiksi kuparijohtimiksi sanotaan kylmä liitokseksi. Kuparijohtimien käyttäminen kylmän pään jälkeen on sallittua, koska ne kumoavat toistensa synnyttämät jännitteet. Standardien ilmoittamat jännitteet termopareille pitävät paikkaansa vain, kun kylmän pään lämpötila on 0 ˚C. Kylmän pään sijoittaminen 0 ˚C tuntumaan on hankalaa ja joissakin paikoissa mahdotonta tai todella kallista, täytyy kylmä pään lämpötilaa kompensoida esim. termistorilla, diodilla tai Pt-100 elementillä, joilla saadaan selville kylmän pään absoluuttinen lämpötila. Tällöin kohteen todellinen lämpötila saadaan lisäämällä kylmän pään lämpötilaa vastaava jännite mitattuun jännitteeseen. Termoparit eivät ole täysin lineaarisia, mutta nykyisin digitaalisiin piireihin on tallennettu termoelementin taulukko muistiin, jolloin lukeman tarkkuus saadaan hyvin lähelle todellista lämpötilaa. (Termopari lämpötila-anturina)

Termoelementtejä on montaa eri tyyppiä eri ominaisuuksille ja ne ovatkin värikoodattuja kylmä liitos pään kaapeleista taulukon 1 mukaan.

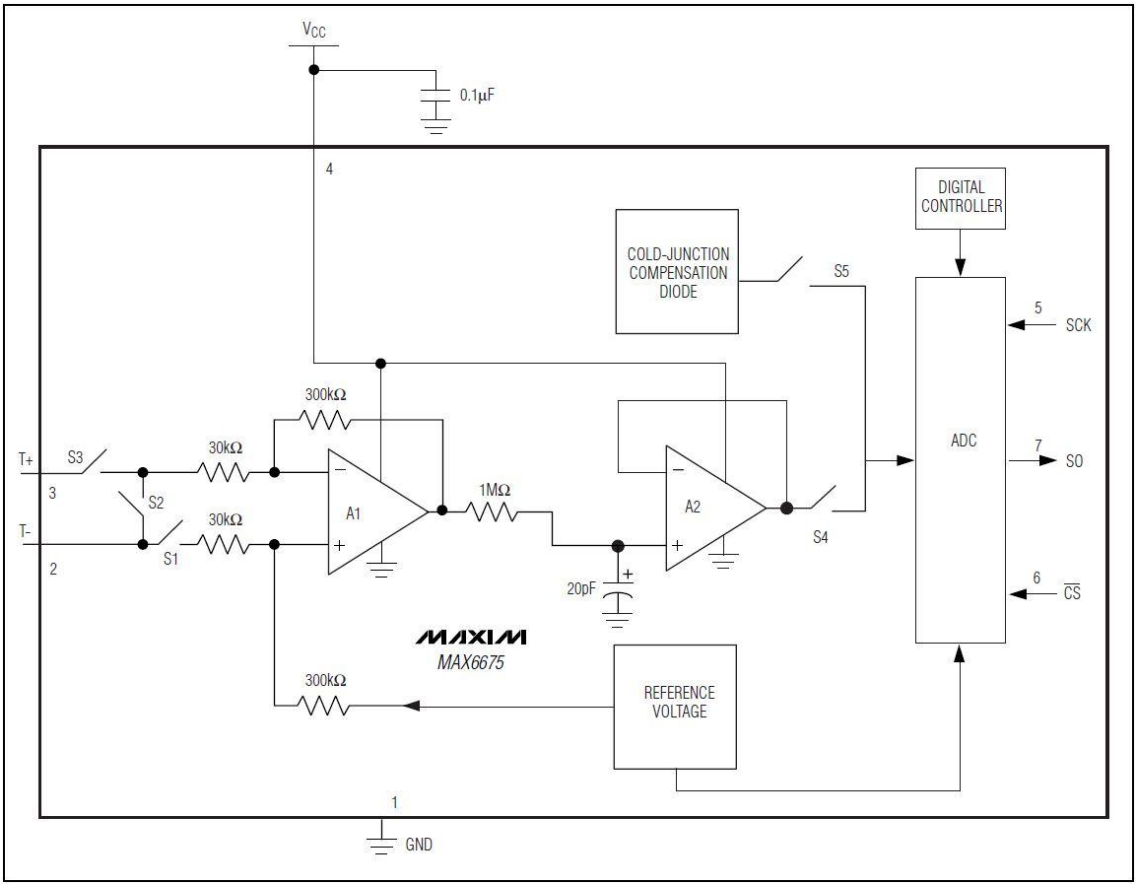
Tyyppi	Materiaali	BS1834	ANSI MC 98.1	DIN 43714	IEC584
U Kompensointi R ja S Tyypeille	+ Kupari - Kupari-Nikkeli				
J	+Rauta -Konstantaani				
K	+ Nikkeli-Kromi - Nikkeli-alumiini				
Vx Kompensointi K tyypille	+Kupari -Konstantaani				
T	+ Kupari -Konstantaani				
E	+Nikkeli-Kromi - Konstantaani				
N	+ Nicrosil - Nisil				

TAULUKKO 1. Tyypit, materiaalit ja niitä vastaavat värikoodit. (Termopari lämpötila-anturina)

2.4.3 Maxim MAX6675 Termopari A/D-muunnin

MAX6675 on K-tyypin termoparille tehty kylmä liitos kompensoitu 12-bittinen A/D -muunnin. Kuuman pään mittausalue on 0 - 1024 ˚C tarkkuudella 0,25 ˚C. Kylmän pään (ympäristön lämötila) lämpötila-alue on huomattavasti pienempi -20 ˚C:sta 85 ˚C:een, koska termoparit mittaavat vain kahden pisteen lämpötila eroa, täytyy MAX6675 sijaita samassa kohtaa kuin termoparin kylmä pää sijaitsee, jotta mahdolliset lämpötila virheet saadaan eliminoitua. (Maxim Integrated MAX6675 datasheet)

MAX6675:en termoparin mittaus tapahtuu siten, että ensin se vahvistaa termoparilta tulevan jännitteen ja poistaa häiriöt, joita johtimiin on saattanut indusoitua matkan varrella. Sitten jännite menee A/D -muuntimelle, jossa on myös kylmän pään lämpötila diodin antama jännite ja vähentää diodin antaman jännitteen termoparin jännitteestä ja muuntaa sen 12-bittiseksi digitaaliseksi arvoksi, jonka se lähettää SPI-väylän kautta mikrokontrollerille. (Maxim Integrated MAX6675 datasheet)



KUVA 16. MAX6675 lohkokaavio. (Maxim Integrated MAX6675 datasheet)

3 TOTEUTUS

3.1 Osien tilaus

Suunitelman valmistuttua oli tarvittavien osien tilauksen aika. Verrattiin useiden kotimaisten sekä ulkomaalaisten verkkokauppojen hintoja ja osien saatavuuksia. Ongelmana oli, että osat olisi pitänyt tilata useammasta eri kaupasta ympäri maailmaa. Näin pelkästään postimaksut olisivat yli kaksinkertaistaneet kustannukset siitä, mitä pelkät komponentit olisivat maksaneet. Tämän takia päätös oli tilata kaikki osat Ebay:stä. Ebay on sivusto, jossa kuka tahansa voi myydä tavaraa. Ebay:n käytön helppous, turvallisuus Paypal:in kautta ja myyjän maineen tarkistaminen ovat sen tärkeimpiä etuja. Myös suurimpaan osaan myytävien tuotteiden hinnoista sisältyy postikulut. Silti komponentit postikuluineen olivat halvemmat, kuin ns. luotettavimmissa kaupoissa.

Osien alkuperä oli suurimmaksi osaksi Kiina. Laadusta ei ole tietoa ja todennäköisesti suurin osa osista ovat kopioita eikä ainakaan Arduino-kehitysalustat ulkonäöstään huolimatta ole tehty Italiassa. Se ei silti ole rikos, sillä Arduino-kehitysalustoja saa tehdä kuka vain ja Arduinon sivuilla onkin Eagle-piirilevykuvat.

3.2 Ohjelma Megalle

Osien saavuttua aloitettiin ohjelmakoodin kirjoittaminen Arduino-kehitysalustoille käyttäen Arduinon omaa IDE-ohjelmaa. Ensimmäisenä ohjelma rakennettiin Megalle, koska tulevaisuudessa siihen saatetaan lisätä uusia moduleita. Tämän takia koko sisällön rakenne on helpompi aloittaa sillä, mihin muutoksia saattaa myöhemmässä vaiheessa tulla. Korttien välinen rajapinta piti saada mahdollisimman helpoksi ja koska suurta vikasietoisuutta ei tässä työssä tarvita, voidaan ohjelmasta karsia suuria määriä varmistuksia ja muita tarkistusfunktioita. Ohjelman koko ja yksinkertaisuus olivat myös tärkeitä kriteereitä. Tulevaisuudessa ohjelman päivittäminen piti onnistua muiltakin kuin työn alkuperäiseltä tekijältä ja ettei ohjelman ymmärtäminen olisi mahdotonta aloittelijalle, käytettiinkin ohjelmassa paljon ulkopuolisia kirjastoja pääohjelman selkeyttämiseksi. Näitä monipuolisia kirjastoja ei tässä työssä keritä käydä läpi.

3.2.1 Käyttöliittymä

Ensimmäisenä etsittiin tietoa SSD1289-ohjainpiiristä, mutta sitä olikin hyvin vähän, koska piiri oli suhteellisen uusi ja harvinaisempi kuin muut vastaavat. Pitkän etsinnän jälkeen löytyi ITDB02_Graph16 niminen kirjasto, jonka oli tehnyt Henning Karlsen. Kirjasto tuki 16-bittistä tiedonsiirtoa, joka on nopeampi kuin 8-bittinen versio. Arduino Megalla ei ollut pulaa pinneistä, joten tämä kirjasto oli oiva valinta tähän työhön. Kirjastossa oli valmiina pienet, isot ja seitsemän segmentti fontit. Yksi erikoismerkki puuttui, mutta sen sai ohjelmaan lisättyä korvaamalla jonkun toisen tarpeettoman merkin. Kirjastossa oli myös tuki yksinkertaisen grafiikan piirtämiselle mm. suorakulmio, viiva, väritys, ympyrä ja paljon muuta.

Kirjaston käyttöönotto oli hyvin helppoa: lisättiin kirjasto ja halutut fontit ohjelman alkuun

```
#include <ITDB02_Graph16.h>
extern uint8_t SmallFont[];
extern uint8_t BigFont[];
```

Näytölle annettiin nimi myGLCD ja sen alustaminen suoritettiin komennolla *InitLCD()*, jossa pystyttiin määrittämään haluttiinko näyttö pystyyn vai vaakaan. Ilman parametriä funktio käyttää alustaa näytön pystyyn. Seuraavana alustettiin näyttö mustaksi puhdistus komennolla *clrScr()* ja valittiin tekstin värit antaen arvo jokaiselle värille asteikolla 0 - 255 komennolla *setColor(punainen, vihreä, sininen)*. Käytetty fontti komennolla *setFont()*, jonka parametreina valittavana oli *SmallFont, BigFont* tai *SevenSegNumFont*.

```
//Setup the LCD
    myGLCD.InitLCD();
    myGLCD.clrScr();
    delay(10);

    //teksti valkoista
     myGLCD.setColor(255,255,255);
```

Näytön alustuksien jälkeen lisättiin muuttumattomat tekstit näyttöön, jotka nekin kuuluivat *setup()* -funktion sisälle. Tekstin kirjoittamiseen käytettiin funktiota *print("haluttu teksti", aloitus x-koordinaatti, aloitus y-koordinaatti)* ja työn nopeuttamiseksi kirjasto tuki sanoja *LEFT, CENTER, RIGHT*, joilla tekstin aloituspistettä x-koordinaatistossa ei tarvitse itse laskea.

```
//Ylärivin tekstit
myGLCD.setFont(SmallFont);
myGLCD.print("Latausteho", LEFT, 1);
myGLCD.print("Savukaasut", RIGHT, 1);

//teho ja Savukaasuojen Lämpötilat (ISO TEKSTI)
myGLCD.setFont(BigFont);
myGLCD.print("kW", 30, 20);
myGLCD.print("~C", RIGHT, 20);
```

Muuttumattomat nuolet ja säiliö piirrettiin käyttäen kirjaston *drawline(X1, Y1, X2, Y2)* -funktiota, jonka X1- ja Y1 -koordinaatit ovat viivan aloituspiste ja X2 sekä Y2 ovat viivan lopetuspiste. Säiliön piirtämiseen käytettiin *drawroundrect(X1, Y1, X2, Y2)* -funktiota, joka piirtää pyöristetyillä kulmilla olevan suorakulmion parametreinä annettavien vasemman yläkulman X1-, Y1 -koordinaatit sekä oikean alakulman X2-, Y2 -koordinaattien välille.

```
//Suunta nuolet (sisään)
myGLCD.drawLine(209, 220, 239, 220);  //-----
myGLCD.drawLine(209, 226, 239, 226);  //_____
myGLCD.drawLine(239, 220, 239, 226);  // |
myGLCD.drawLine(209, 220, 209, 215);  // nuolen ylempi |
myGLCD.drawLine(209, 226, 209, 231);  // nuolen alempi |
myGLCD.drawLine(209, 215, 190, 223);  // Nuolen kärjen ylempi viiva
myGLCD.drawLine(209, 231, 190, 223);  // Nuolen kärjen alempi viiva

//säiliö
myGLCD.drawRoundRect(80, 190, 160, 300);
```

Näytön muuttumattomien piirtämisen jälkeen siirrytään pois *setup()* -funktiosta lopulliseen *loop()* -funktioon, jossa ajettiin lämpötila-arvojen ja tehon arvon piirtäminen ja päivittäminen aina tarpeen tullen. Näytölle piirrettäessä täytyi ottaa huomioon

lämpötila-arvojen ja tehon arvon kerrannaisyksiköiden määrä. Näytölle kirjoitettaessa täytyy ottaa huomioon uudelleenkirjoitus, jolloin vanha arvo pitää "putsata" uuden tieltä. Savukaasujen lämpötila saattoi vaihdella 0 ˚C - 600 ˚C välillä ja tehon arvot 0 kW - 50 kW välillä, joten näytölle piirtäessä ongelmana oli vaikkapa 100 ˚C siirtyminen näyttämään 99 ˚C, jolloin ensimmäinen numero täytyi kirjoittaa tyhjäksi, ettei se jäisi näytölle kummittelemaan ja näyttämään lämpötilaksi 199 ˚C. Ongelma kierrettiin yksinkertaisilla if-lauseilla, joissa rajattiin arvot lämpötiloille 0 - 9, 10 - 99 ja 100 - 999 ja piirrettiin tarvittaessa tyhjää ennen lukuarvoa vanhan luvun päälle.

```
if(savuL < 10){                          //savulämpöjen vanhojen arvojen poisto näytöstä.
   myGLCD.print("  ", 159, 20);           // | putsataan ylimääräiset numeropaikat
   myGLCD.printNumI(savuL, 191, 20);     // |
}                                        // |
                                         // |
if(savuL < 100 && savuL > 9){            // |
   myGLCD.print(" ", 159, 20);           // |
   myGLCD.printNumI(savuL, 175, 20);     // |
}                                        // |
                                         // |
if(savuL >= 100 && savuL <=999){         // |
   myGLCD.printNumI(savuL, 159, 20);     // | yli 999 asteen lämpötilat hyödyttömiä.
}                                        // x
```

Viimeisenä näytölle piirrettiin säiliön ylä- ja alalämpötilat sekä latausvesien lämpötilat. Säiliön ja latausvesien lämpötilat eivät koskaan ylitä 100 ˚C eivätkä laske alle 10 ˚C, joten ne eivät tarvitse savukaasuissa ja tehossa olevaa korjausta lämpötila-arvon piirtämiseen. Käyttöliittymästä saatiin hyvinkin suunnitelman kaltainen.

KUVA 17. Valmis käyttöliittymä.

3.2.2 Lataustehon laskeminen

Tehoalifunktion *tehoAlifunktio(tulevan veden lämpötila, lähtevän veden lämpötila)* tehtävänä oli laskea lämmityskattilalle menevän ja tulevan veden lämpötilaerosta latausteho. Lataustehon määrittämiseksi vaadittiin kiertoveden virtaus kuutioina tunnissa, veden ominaislämpökapasiteetti (4,2kJ/kgC˚) ja veden tiheys (1000kg/m^3). Kaksi jälkimmäistä tiedettiin varmasti, mutta ensimmäinen, eli virtausmäärä, oli mahdotonta selvittää ilman erillistä virtausmittaria. Arvioitu virtaus laskettiin jakamalla kahdella pumpun kyljessä ilmoitettu kuormittamaton virtausmäärä, josta saadaan tulokseksi kolme kuutiota tunnissa. Testien perusteella tehon mittaus näyttäisi olevan tarpeeksi lähellä todellista lukemaa ja tulevaisuudessa järjestelmän seuraavan tyhjennyksen yhteydessä lisätään siihen myös virtausmittari tarkemman arvon saamiseksi. (Talukot.com)

3.2.3 Tilat ja sarjaliikenne

Datan kerääminen Unolta jaoteltiin eri tiloihin selkeyden, nopeuden ja vikaherkkyyden takia. Jonkin arvon saapumatta jääminen sekoittaisi koko hakuprosessin ja kortti olisi käynnistettävä uudelleen. Eri lämpötila-antureita oli viisi kappaletta ja jokaiselle oli oma tilansa. Ensimmäisenä otettiin vanha arvo talteen, koska uutta arvoa ei ihan heti saada. Mega käy läpi ohjelmansa huomattavasti nopeammin kuin Uno, jolloin voidaan tilarakenteella mahdollistaa muiden ohjelmaosioiden ajaminen sillä välin, kun Uno pyytää lämpötilaa lämpötila-anturilta ja lähettää sen Megalle. Uuden arvon saatuaan, nollattiin uusiintalähetyslaskuri tarpeettomana ja siirryttiin seuraavaan tilaan. Uusintalähetyslaskuri pitää huolen siitä, ettei Mega jää jumiin yhteen tilaan kutsun hukuttua häiriöihin. Kaikkien tilojen käytyä läpi, voitiin palata takaisin ensimmäiseen tilaan. Ohjelmaan on helppo lisätä uusia tiloja, koska seriali alifunktio hoitaa uuden arvon päivittämisen vanhan tilalle sen saapuessa Unolta.

3.3 Ohjelma Unolle

Unon ohjelmoinnissa sovellettiin samoja periaatteita kuin Megan ohjelmoinnisa. Modulaarisuus ja yksinkertaisuus helpottavat mahdollisia lisäyksiä myöhemmässä vaiheessa sekä aloittelijalle soveltuva yksinkertaisuus mahdollistaa myöhemmät kehityslisäykset muiltakin kuin työn alkuperäiseltä tekijältä. Prosessoriajan käyttö maksimoidaan tekemällä lämpötilanmittaukset vain niitä pyydettäessä, jotta muille kuin lämpötilanmittauksille riittäisi myös prosessointiaikaa. Prosessointiaikaa ei vielä tässä vaiheesssa ainakaan hyödynnetty, mutta myöhemmässä vaiheessa saattaa olla, että kortilla tehdään muutakin kuin lähetetään tietoja eteenpäin. Unon ohjelmistossa käytetään kahta kirjastoa ohjaamaan Dallasin lämpötila-antureita ja Maximin termoparin A/D -muunninta.

3.3.1 Alustukset

Alustuksessa ensimmäisenä oli lämpötila-anturien kytkentäpinni ja anturien tarkkuuden valinta. Dallas anturien vakio tarkkuus oli työhön suhteettoman tarkka ja hidas 12-bittinen, joten se asetettiin huomattavasti nopeammaksi 9-bittiseksi.

```
#define ONE_WIRE_BUS 2                  //OneWire käyttää pinniä 2
#define TEMPERATURE_PRECISION 9         //Lämpötila-anturien tarkkuus 9-bittiä
```

Kääntäjäkohtaisten käskyjen jälkeen oli globaalit muuttujat *TempC, tila* ja *lampotila*, jotka helpottivat koodin lukua sekä toimivat apumuuttujina alifunktioissa että switch case -rakenteessa. Maximin pinnien paikat löytyvät myös globaaleista muuttujista nimillä *CS, SO, SCKO* ja lämpötilan näyttömuoto nimellä *units*.

DeviceAdress on talulukko, jossa säilytettiin jokaisen lämpötila-anturin osoitteita ja nimiä. Osoitteet saatiin antureista kytkemällä ne yksitellen koekytkentälevylle ja ajamalla *oneWire.Search(tallennettavan osoitetaulukko)* ja tulostamalla se sarjaliikenne monitoriin, josta osoite voitiin lukea.

Seuraavana kerrottiin kirjastoille *oneWire*, *DallasTemperature* ja *MAX6675* niiden kytkentäpinnit, jonka jälkeen voitiin käynnistää *DallasTemperature* ja asettaa lämpötila-

anturien resoluutiot 9-bittisiksi. Lopuksi sarjaliikenneväylä käynnistettiin toimimaan Megan kanssa samalla nopeudella 2400 bps.

3.3.2 Lämpötilanmittaus

Lämpötilanmittaukset suoritettiin jokainen omassa lähes identtisessä alifunktiossaan ja eroavatkin toisistaan vain nimensä, lämpötila-anturinsa sekä kertoimensa puolesta. Jokaisessa alifunktiossa käytettiin kolmea apumuuttujaa, jotka nimettiin *i, keskiarvo* ja *palautus*. *DallasTemperature* kirjasto antaa mittaustuloksen liukulukuna, jolloin tarvitaan tulokselle sitä varten float tyyppinen muuttuja *keskiarvo*. Mittaustulos haetaan kaksi kertaa tarkkuuden lisäämiseksi ja jotta vältytään virhetilanteessa ilmoitetulta -127 ˚C tulokselta. Lopuksi ennen tuloksen palauttamista tehdään tulokselle tyyppimuunnos floatista integeriin, koska näyttö ei näytä desimaalilukuja ja virhemarginaalit ovat muutaman asteen luokkaa, jolloin tarkemmasta tiedosta ei hyödytä mitään.

```
int lahtevaKeskiarvo()
{
   int i;
   float keskiarvo = 0;
   int palautus = 0;

   for (i = 0; i < 2; i++) {
      sensors.requestTemperatures(); // Pyydetään lämpötilan mittausta
      tempC = sensors.getTempC(lahtevaS) * 1.05;
      keskiarvo = keskiarvo + tempC;  //keskiarvolaskentaa dallasin lämpötilasta
   }
   keskiarvo /= 2;
   palautus = (int) keskiarvo;

   return palautus;
}
```

Jokaisella lämpötila-anturilla on korjauskerroin. *lahtevaKeskiarvo()* -alifunktiolla se oli 1,05. Koska lämpötila-anturia ei ollut mahdollista asentaa mekaanisille mittareille varattuihin putkiin, täytyi anturit asentaa säiliön vaipan alle missä lämpötila oli alhaisempi kuin säiliön vesitilassa. Jatkossa korjauskertoimia säädetään vastaamaan paremmin todellista lämpötilaa hienosäätämällä kertoimia.

3.3.3 Pääohjelma

Pääohjelmassa prosessoriaikaa haluttiin säästää mahdollisimman paljon muille tulevaisuudessa lisättäville tehtäville ja vain tarpeen tullen käytäisiin mittaamassa haluttu lämpötila. Samalla ohjelmasta tehtiin myös helppolukuinen ja yksinkertainen käyttämällä Switch case -rakennetta.

```
if(Serial.available() > 0){              //luetaan onko sarjaliikenneväylässä pyyntöä
  tila = Serial.read();                  //jos on, niin tallennetaan lähete muuttujaan tila
```

Uno kuuntelee Sarjaliikenneväylää ja sinne saapuvaa tietoa seuraavasta lämpötilapyynnöstä. Pyynnön saavuttua tallennettiin tieto muuttujaan *tila*, jota vertailtiin eri tapauksiin.

```
case 1:
  lampotila = tulevaKeskiarvo();      //varaajaan tulevan veden lämpötila
  Serial.write(lampotila);
  lampotila = 0;
  break;
```

Esimerkiksi jos *tila* sisälsi arvon yksi, valittiin sitä vastaava tapaus ja suoritettiin *tulevaKeskiarvo()* -alifunktio, lähetettiin tulos sarjaliikenneväylään, nollattiin muuttuja ja lopetetaan Switch case -rakenteen ajaminen *break;* -komentoon siirtymällä takaisin pääohjelmaan.

4 JATKOKEHITYSMAHDOLLISUUDET

4.1 Aurinkokeräimet

Tarkoituksena olisi lisätä aurinkokeräinten seuranta ja ohjaus yksinkertaisella logiikalla, joka vertailee säiliön alalämpötilaa ja lämpökeräinten lämpötilaa. Lämpökeräinten lämpötilan ylittäessä säiliön lämpötilan 5 ˚C:lla, käynnistyisivät pumput, jotka kierrättävät vettä lämpökeräimissä niin kauan kunnes lämpötilaero säiliön ja keräinten välillä olisi enää 2 ˚C. Näin saataisiin maksimaalinen lämmitysteho keräimillä ja vältytäisiin jatkuvalta pumppuja rasittavalta päälle- ja poiskytkemiseltä. Seurantajärjestelmän rinnalle tulee myös asentaa varajärjestelmä, joka häiriötilanteessa laittaa pumput päälle, jos keräinten linjassa lämpötila kasvaa liian suureksi.

4.2 Sähkönkulutus

Sähkönkulutuksen mittaus voidaan toteuttaa yksinkertaisella valoherkällä diodilla, joka mittaa sähkökaapissa olevan mittarin valopulssien määrää ja lasketaan siitä hetkellisen tai pidemmän ajan kulutuksen ja ilmoittaa sen näytöllä joko kuvaajana tai pelkkänä numeroarvona. Kuvaajan käyttöä varten näyttöön voidaan lisätä työpöytiä, joita voidaan ohjata kosketuksen avulla.

4.3 WEB-käyttö

Tuloksia voitaisiin tarkastella myös netin kautta. Tätä varten tarvittaisiin Arduinon ethernet shield W5100 tai jokin muu vastaava piiri, jolla seurantajärjestelmä pääsisi internettiin. Tarkkaa tietoa tämän toiminnallisuuden lisäämisestä ei ole, mutta ainakin yksinkertainen nettisivu voitaisiin jo tällä kokoonpanolla tehdä. Monimutkaisempaan todennäköisesti vaadittaisiin joko vuokrattu serveri tai jokin muu vastaava, mihin Arduinolta lähetettäisiin tietoa tietyn aikajakson välein. Serverillä sitten olisi tietokanta datasta pitemmältä aikaväliltä, josta voitaisiin tehdä kuvaajat ja näyttää muita haluttuja tietoja.

4.4 GSM-ohjaus ja hälytys

GSM-yhteyden saamiseksi vaaditaan GSM-moduli. Esimerkkinä Simcom 900D, jolla voitaisiin lisätä mm. etäohjaus, lämpötila-arvojen etäkysely, toimintahäiriöiden ja raja-arvojen ylityksien ilmoitukset ja asettaminen jne. Kaikkiin etäohjaustoimintoihin tottakai vaadittaisiin jokin tunnistautumiskoodi esimerksiksi pin-koodi, etteivät mainokset ja muut vastaavat tekstiviestit laukaisisi tietojen lähettämistä.

4.5 Kodinseuranta

Seurannan lisäämiseksi voitaisiin toinen etäyksikkö asentaa sisätiloihin, jossa olisi lämpötila-antureita huoneissa lämpötilan datan keruuta varten sekä liikettunnistimia, jotka aiheuttaisivat hälytyksen havaitessaan liikettä silloin kun talossa ei kuuluisi olla ketään.

4.6 Piirilevyt

Kaikki kytkennät ovat tehty koekytkentälevyille ja liitetty toisiinsa käyttäen johtoihin kolvattuja piikkirimoja. Järjestelmän varmuuden lisäämiseksi ja selkeyttämiseksi sekä yhdistämiseksi eri komponentteja pienempään tilaan voitaisiin piirilevyt piirtää itse ja kolvata komponentit paikoilleen. Näin saataisiin järjestelmä häiriövarmemmaksi ja yleisesti helpommaksi korjata ja käsitellä.

5 YHTEENVETO

5.1 Työn tulokset ja toteutuminen

Tämän työn tavoitteena oli lämmityskeskuksen lämpötilanseurannan helpottaminen. Työ saatiin vaatimuksiin nähden valmiiksi, vaikkakin kehitettävää vielä onkin korjausarvojen hienosäädön, piirilevyjen ja johdotusten osalta, mutta järjestelmä toimii tälläisenäänkin riittävän hyvin käyttötarkoituksessaan. Työtä ei voi suositella käytettäväksi mihinkään pääasiallisena ohjausjärjestelmänä sen herkän vikaantumisen ja muutamien muiden puutteiden takia. Työssä käytettiin paljon muiden tekemiä kirjastoja, jotka helpottivat ja nopeuttivat sen tekemistä, mutta joiden eheydestä ja toimivuudesta ei aina voi olla varma.

5.2 Pohdinta

Työ oli itsessään kiinnostava ja palkitseva sekä antoi paljon uutta tietoa ja ideoita jatkokehitystä varten. Työssä käytettiin paljon valmiita kirjastoja ja niitä joutui tutkimaan ja hiukan muokkaamaankin sopimaan käyttötarkoitukseensa. Raportissa kirjastoja ei läpi käyty niiden laajuuden takiaa. Järjestelmän suunnitelussa täytyi ottaa huomioon paljolti ohjelmakoodin lukemisen helppous. Monet kohdat olisi voinut kirjoittaa lyhyemmin ja yhdistellä eri funktioita isommiksi kokonaisuuksisi ja yleisesti käyttää enemmän luovuutta ja ideointia. Lopputulos on silti toimiva ja jatkoa varten helposti muokattavissa, joten korjattavaa siltä osin työssä ei ole.

Järjestelmää testattiin aluksi pelkästään koekytkentälevyllä, jossa se toimi moitteetta. Koekytkentöjen jälkeen laitteisto kasattiin niille kuuluville paikoille ja siitä ongelmat alkoivat. Aluksi mikään ei toiminut. Vikojen etsimiseen ja korjaamiseen aikaa menikin huomattavasti kauemmin kuin olisi alunperin olettanut. Suunnitelman ei ollut tarkoitus olla täydellinen ja siihen tulikin paljon muutoksia projektin edetessä. Työtä tehdessä huomasi kuinka paljon suunnitelmista puuttui yksityiskohtia, jotka olivat projektin onnistumisen kannalta tärkeitä.

LÄHTEET

Jämsä, L. 2010. Arduino-alustat esitelyssä. Luettu 29.3.2012.
http://www.ruuvipenkki.fi/2010/08/12/arduino-alustat-esittelyssa

Ariterm. Arimax puulämmitys. Luettu 4.4.2012.
http://195.67.82.150/ariterm/Puulammitys%20low%20res.pdf

1-Wire -Tekniikka. Luettu 26.4.2012.
http://www.nic.fi/~skarna/etusivu.html#kuvat

Signaalinkäsittelytekniikan laboratorio. 2003. Luettu 4.5.2012.
http://signal.hut.fi/digis/luento1/tieto.html

Benjamin New. 2007. Considerable Sounds: Analog Vs. Digital Circuitry - Which Is Better For Music. Luettu 4.5.2012.
http://dulyconsider.blogspot.com/2007/10/considerable-sounds-analog-vsdigital_23.html

Wikipedia. 2012. RS232. Luettu 7.5.2012.
http://en.wikipedia.org/wiki/RS-232

Lammert Bies. 2011. RS232 Specifications and standard. Luettu 7.5.2012.
http://www.lammertbies.nl/comm/info/RS-232_specs.html

H, Honkanen. Anturit. Luettu 7.5.2012.
http://gallia.kajak.fi/opmateriaalit/yleinen/honHar/ma/ELE_A%20N%20T%20U%20R%20I%20T.pdf

fpga4fun. What is SPI?. Luettu 31.7.2012.
http://www.fpga4fun.com/SPI1.html

Asko, Olli. 2010. Sulautettujen järjestelmien väylätekniikat. Luettu 31.7.2012.
http://publications.theseus.fi/bitstream/handle/10024/13265/opinnaytetyo.pdf?sequence=1

Semtu. VEMO-valuankkurit. Käyttöohje. Luettu 18.3.2011.
http://www.semtu.fi/?file=240

Maxim Integrated Products Inc 2012 profile. Luettu 16.11.2012
http://www.maximintegrated.com/company/profile.cfm

Maxim Integrated DS18B20 datasheet. Luettu 11.1.2013
http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

Dallas Semiconductor DS1820. Luettu 11.1.2013
http://www.micropik.com/PDF/ds1820.pdf

Pietiko Oy 021209. Termopareista. Termopari lämpötila-anturina. Luettu 26.1.2013
http://www.pietiko.fi/pietiko/sovellus/Termopari.pdf

Maxim Integrated MAX6675 datasheet. Luettu 28.1.2013
http://datasheets.maximintegrated.com/en/ds/MAX6675.pdf

Henning Karlsen. Library: ITDB02_Graph16. Luettu 21.3.2012
http://www.henningkarlsen.com/electronics/library.php?id=39

Talukot.com. Luettu 19.2.2013
http://www.taulukot.com/index.php?search_id=mekaniikka_termodynamiikka&lng=fi#nesteiden_ominaisuuksia

Matthew Ford. 2010. Bluetooth Controlled Led Driver. Luettu 19.2.2013
http://www.forward.com.au/BluetoothLedDriver/BluetoothControlledLedDriver.html

LIITTEET

Liite 1. Arduino Megan lähdekoodi

1(5)

```
#include <ITDB02_Graph16.h>
#include <avr/pgmspace.h>

extern uint8_t SmallFont[];
extern uint8_t BigFont[];

ITDB02 myGLCD(38,39,40,41,ITDB32S);   //näytön asetus-arvot (RS,WR,CS,RST, malli)

  int teho = 0;
  int savuL = 0;
  int ylaLampo = 0;
  int alaLampo = 0;
  int lahteva = 0;
  int tuleva = 0;
  int tila = 0;
  int vanhaArvo = 0;
  long uusiLahetys = 0;
  boolean eka = true;


void setup()
{
      randomSeed(analogRead(0));
     Serial1.begin(2400);

     //Setup the LCD
    myGLCD.InitLCD();
    myGLCD.clrScr();
    delay(10);

    //teksti valkoista
     myGLCD.setColor(255,255,255);

     //Ylärivin tekstit
     myGLCD.setFont(SmallFont);
```

```
myGLCD.print("Latausteho", LEFT, 1);

myGLCD.setFont(SmallFont);
myGLCD.print("Savukaasut", RIGHT, 1);

//teho ja Savukaasuojen Lämpötilat (ISO TEKSTI)
myGLCD.setFont(BigFont);
myGLCD.print("kW", 30, 20);
myGLCD.print("~C", RIGHT, 20);

//Säiliön lämpötilat näyttölle
myGLCD.setFont(SmallFont);
myGLCD.print("Lampotilat", LEFT, 170);

myGLCD.print("~C", 22, 190);
myGLCD.print("~C", 22, 280);

//meno/paluu-vesien lämpötilat
myGLCD.print("Latausvesi",RIGHT,170);
myGLCD.print("~C", 223, 190);
myGLCD.print("~C", 223, 280);

//Suunta nuolet (sisään)
myGLCD.drawLine(209, 220, 239, 220);  //-----
myGLCD.drawLine(209, 226, 239, 226);  //_____
myGLCD.drawLine(239, 220, 239, 226);  // |
myGLCD.drawLine(209, 220, 209, 215);  // nuolen ylempi |
myGLCD.drawLine(209, 226, 209, 231);  // nuolen alempi |
myGLCD.drawLine(209, 215, 190, 223);  // Nuolen kärjen ylempi viiva
myGLCD.drawLine(209, 231, 190, 223);  // Nuolen kärjen alempi viiva

//Suuntanuolet (ulos)
myGLCD.drawLine(190, 258, 220, 258);  //-----
myGLCD.drawLine(190, 264, 220, 264);  //_____
myGLCD.drawLine(190, 258, 190, 264);  // |
myGLCD.drawLine(220, 258, 220, 253);  // nuolen ylempi |
myGLCD.drawLine(220, 264, 220, 269);  // nuolen alempi |
myGLCD.drawLine(220, 253, 239, 261);  // Nuolen kärjen ylempi viiva
myGLCD.drawLine(220, 269, 239, 261);  // Nuolen kärjen alempi viiva
```

```
    //säiliö
    myGLCD.drawRoundRect(80, 190, 160, 300);
    myGLCD.print("SAILIO", CENTER, 220);
    myGLCD.print("3000L", CENTER, 240);


  }

int tehoAlifunktio(int tuleva, int lahteva){
     long int apu;
     if( lahteva >=50)
     {
        apu = tuleva - lahteva;
        apu *= 3;                          //KIERTOVEDEN KUUTIOMÄÄRÄ ARVIO!
        apu *= 4200;                       //veden ominaislämpökapasiteetti (4,2kJ/kgC)
        apu *= 1000;                       //veden tiheys(1000kg/m^3)
        apu /= 3600000;                    //yksikkömuunnoskerroin (kj -> kWh)
     }
     else apu = 0;
     return apu;
}

int seriali( int anturi, int vanha)
{
    if(eka || uusiLahetys == 1000000){   //onko ensimmäinen kerta tässä tilassa? jos on niin lähetetään
                        // lämpötilapyyntö. Jos tulosta ei kuulu,  niin uusi lähetetään muutaman sekunnin päästä.
    Serial1.write(anturi);                  //pyyntö
    eka = false;            //muutetaan tila falseksi, että ei tulla tänne heti uudestaan ja lähetetä uutta pyyntöä
    uusiLahetys = 0;
    }
    uusiLahetys++;
    if(Serial1.available() > 0){         //kun serialiin on tullut jotakin
       int palautus;
       palautus = Serial1.read();
       tila += 1;                         //Siirrytään seuraavaan tilaan, eli anturiin
       if(tila == 5){                     //Jos ollaan viimeisessä anturissa, aloitetaan alusta.
        tila = 0;
      }
    eka = true;
    return palautus;
   }
```

```
 else
  return vanha
}

void loop()
{

  //teho ja Savukaasuojen Lämpötilat (ISO TEKSTI)
  myGLCD.setFont(BigFont);
  if(teho >= 0 && teho < 100)       //estetään näyttöä sekoamasta, eikä tälläisiin lataustehoihin edes päästä.
  {
      if(teho <  10){                           //estetään nollan alapuolelle menemästä
        myGLCD.print(" ", LEFT, 20);          //tyhjennetään mahdolliset numerot
        myGLCD.printNumI(teho, 16, 20);    //syötetään <10 numero
      }
      else
      myGLCD.printNumI(teho, LEFT, 20);    //syötetään kaksinumeroinen
  }
   else
   myGLCD.print("0", 16, 20);           //miinusmerkkisen tilalle annetaan nollaa


  if(savuL < 10){                            //savulämpöjen vanhojen arvojen poisto näytöstä.
     myGLCD.print("  ", 159, 20);             // | putsataan ylimääräiset numeropaikat
     myGLCD.printNumI(savuL, 191, 20);   // |
  }                                                 // |
                                                    // |
  if(savuL < 100 && savuL > 9){             // |
     myGLCD.print(" ", 159, 20);              // |
     myGLCD.printNumI(savuL, 175, 20);   // |
  }                                                 // |
                                                    // |
  if(savuL >= 100 && savuL <=999){        // |
     myGLCD.printNumI(savuL, 159, 20);   // | yli 999 asteen lämpötilat hyödyttömiä.
  }                                                 // x

  //Säiliön lämpötilat näyttölle
  myGLCD.setFont(SmallFont);
  myGLCD.printNumI(ylaLampo, 4, 190);
  myGLCD.printNumI(alaLampo, 4, 280);
```

```
//lataus meno/paluu-vesien lämpötilat
 myGLCD.printNumI(tuleva, 206, 190);
 myGLCD.printNumI(lahteva, 206, 280);

 teho = tehoAlifunktio(tuleva, lahteva);


 if(tila == 0){                          //tila, eli anturi jonka arvo halutaan
    vanhaArvo = lahteva;
    lahteva = seriali(tila, vanhaArvo);
    if(lahteva >=100 || lahteva <= 10){
       lahteva = 0;
    }
  }
 if(tila == 1){
    vanhaArvo = tuleva;
    tuleva = seriali(tila, vanhaArvo);
    if(tuleva >=100 || tuleva <= 10){
       tuleva = 0;
    }
 }

 if(tila == 2){
    vanhaArvo = ylaLampo;
    ylaLampo = seriali(tila, vanhaArvo);
    if(ylaLampo >=100 || ylaLampo <= 10){
       ylaLampo = 0;
    }
 }
 if(tila == 3){
    vanhaArvo = alaLampo;
    alaLampo = seriali(tila, vanhaArvo);
    if(alaLampo >=100 || alaLampo <= 10){
       alaLampo = 0;
    }
 }
 if(tila == 4){
    vanhaArvo = savuL;
    savuL = seriali(tila, vanhaArvo);
  }
}
```

Liite 2. Arduino Unon lähdekoodi


```
#include <avr/pgmspace.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <MAX6675.h>



#define ONE_WIRE_BUS 2                    //OneWire käyttää pinniä 2
#define TEMPERATURE_PRECISION 9           //Lämpötila-anturien tarkkuus 9-bittiä

float tempC = 0;
int lampotila = 0;
int tila = 0;
int CS = 10;            // CS pinni MAX6675
int SO = 12;             // SO pinni MAX6675
int SCKO = 13;           // SCK pinni MAX6675
int units = 1;          // lämpötilan näyttömuoto (0 = raakadata, 1 = ˚C, 2 = ˚F)


DeviceAddress lahtevaS = { 0x28, 0x07, 0xBA, 0xB6, 0x03, 0x00, 0x00, 0x15 }; //varaajasta lähtevä
DeviceAddress tulevaS = { 0x28, 0xC3, 0xCE, 0xB6, 0x03, 0x00, 0x00, 0x24 };  //Varaajaan tuleva
DeviceAddress ylaS = { 0x28, 0xAB, 0xB5, 0xB6, 0x03, 0x00, 0x00, 0x5B };     //varaaja ylä
DeviceAddress alaS = { 0x28, 0x4F, 0xEA, 0xB6, 0x03, 0x00, 0x00, 0x54 };     //varaaja ala

OneWire oneWire(ONE_WIRE_BUS);        //OneWire nimetään ja annetaan pinni
DallasTemperature sensors(&oneWire);  // dallas käyttämään onewireä
MAX6675 temp(CS,SO,SCKO,units);       // MAX6675 pinnien alustukset

void setup(void)
{
   sensors.begin();
   Serial.begin(2400);

  //asetetaan tarkkuus 9 bittiin
   sensors.setResolution(lahtevaS, TEMPERATURE_PRECISION);
   sensors.setResolution(tulevaS, TEMPERATURE_PRECISION);
   sensors.setResolution(ylaS, TEMPERATURE_PRECISION);
   sensors.setResolution(alaS, TEMPERATURE_PRECISION);
}
```

```
int lahtevaKeskiarvo()
{
   int i;
   float keskiarvo = 0;
   int palautus = 0;

   for (i = 0; i < 2; i++) {
      sensors.requestTemperatures(); // Pyydetään lämpötilan mittausta
      tempC = sensors.getTempC(lahtevaS) * 1.05;
      keskiarvo = keskiarvo + tempC;  //keskiarvolaskentaa dallasin lämpötilasta
   }
   keskiarvo /= 2;
   palautus = (int) keskiarvo;

   return palautus;
}


int tulevaKeskiarvo()
{
   int i;
   float keskiarvo = 0;
   int palautus = 0;

   for (i = 0; i < 2; i++) {
      sensors.requestTemperatures(); // Pyydetään lämpötilan mittausta
      tempC = sensors.getTempC(tulevaS) * 1.09;
      keskiarvo = keskiarvo + tempC;  //keskiarvolaskentaa dallasin lämpötilasta
   }
   keskiarvo /= 2;
   palautus = (int) keskiarvo;

   return palautus;
}


int savuLKeskiarvo()
{
   int i;
   float keskiarvo = 0;
```

```
    for (i = 0; i < 2; i++) {
        tempC = (int)temp.read_temp();
        keskiarvo = keskiarvo + tempC;  //keskiarvolaskentaa savukaasuanturin lämpötilasta
    }
    keskiarvo /= 2;
    return keskiarvo;
}


int ylaLampoKeskiarvo()
{
    int i;
    float keskiarvo = 0;
    int palautus = 0;

    for (i = 0; i < 2; i++) {
        sensors.requestTemperatures(); // Pyydetään lämpötilan mittausta
        tempC = sensors.getTempC(ylaS) * 1.82;
        keskiarvo = keskiarvo + tempC;  //keskiarvolaskentaa dallasin lämpötilasta
    }
    keskiarvo /= 2;
    palautus = (int) keskiarvo;
    return palautus;
}


int alaLampoKeskiarvo()
{
    int i;
    float keskiarvo = 0;
    int palautus = 0;
    for (i = 0; i < 2; i++) {
        sensors.requestTemperatures(); // Pyydetään lämpötilan mittausta
        tempC = sensors.getTempC(alaS) * 1.5;
        keskiarvo = keskiarvo + tempC;  //keskiarvolaskentaa dallasin lämpötilasta
    }
    keskiarvo /= 2;
    palautus = (int) keskiarvo;

    return palautus;
}
```

```
void loop(void)
{
   if(Serial.available() > 0){               //luetaan onko sarjaliikenneväylässä pyyntöä
      tila = Serial.read();                  //jos on, niin tallennetaan lähete muuttujaan tila
      switch (tila){                         //käydään switch case rakenteella läpi vaihtoehdot

         case 0:                             //varaajasta lähtevän veden lämpötila
           lampotila = lahtevaKeskiarvo();       //luetaan arvo anturilta
           Serial.write(lampotila);              //kirjoitetaan arvo sarjaliikenneväylään
           lampotila = 0;                        //nollataan apumuuttuja
           break;                                //katkaistaan switch case ajo ja palataan looppiin

         case 1:
           lampotila = tulevaKeskiarvo();        //varaajaan tulevan veden lämpötila
           Serial.write(lampotila);
           lampotila = 0;
           break;

         case 2:
           lampotila = ylaLampoKeskiarvo();      //säiliön ylempi lämpötila-anturi
           Serial.write(lampotila);
           lampotila = 0;
           break;

         case 3:
           lampotila = alaLampoKeskiarvo();      //säiliön alempi lämpötila-anturi
           Serial.write(lampotila);
           lampotila = 0;
           break;

         case 4:
           lampotila = savuLKeskiarvo();         //savukaasujen lämpötila
           Serial.write(lampotila);
           lampotila = 0;
           break;
      }
   }
```

Liite 3. Kytkentäkaavio